

# An Efficient Parallel Algorithm for Spectral Sparsification of Laplacian and SDDM Matrix Polynomials

Gorav Jindal      Pavel Kolev\*

Max-Planck-Institut für Informatik, Saarbrücken, Germany  
 {gjindal,pkolev}@mpi-inf.mpg.de

## Abstract

A mixture of discrete Binomial distributions (MDBD), denoted by  $\mathcal{B}_{N,T}(\alpha, p)$ , is a set of pairs  $\{(B(p_i, N), \alpha_i)\}_{i=1}^T$ , where  $B(\cdot, \cdot)$  denotes the Binomial distribution, all  $p_i \in (0, 1)$  are distinct,  $\sum_{i=1}^T \alpha_i \leq 1$  and all  $\alpha_i \in (0, 1)$ . A vector  $\gamma$  is induced by MDBD if  $\gamma_i = \sum_{j=1}^T \alpha_j \cdot B_{N,i}(p_j)$  for all  $i \in [0 : N]$ , where  $B_{N,i}(p) = \binom{N}{i} p^i (1-p)^{N-i}$ .

We prove for “large” class  $\mathcal{C}$  of continuous probability density functions (p.d.f.), that for every  $w \in \mathcal{C}$  there exists MDBD with  $T \geq N\sqrt{\phi_w/\delta}$  that  $\delta$ -approximates a *discretized* p.d.f.  $\hat{w}(i/N) \triangleq w(i/N)/[\sum_{\ell=0}^N w(\ell/N)]$  for all  $i \in [3 : N-3]$ , where  $\phi_w \geq \max_{x \in [0,1]} |w(x)|$ . Moreover, we propose an efficient parallel algorithm that on input p.d.f.  $w \in \mathcal{C}$  and parameter  $\delta > 0$ , outputs MDBD that induces a vector  $\gamma$  which  $\delta$ -approximates  $\hat{w}$ . Also, we give an efficient parallel algorithm that on input a discretized p.d.f.  $\hat{w}$  induced by MDBD with  $T = N + 1$  and the corresponding vector  $p$ , outputs exactly the coefficients  $\alpha$ .

Cheng et al. [4] proposed the first sequential algorithm that on input a discretized p.d.f.  $\beta$ ,  $B = D - M$  that is either Laplacian or SDDM matrix and parameter  $\varepsilon \in (0, 1)$ , outputs in time  $\hat{O}(\varepsilon^{-2}mN^2)$ <sup>1</sup> a spectral sparsifier of a matrix-polynomial  $D - \hat{M}_N \approx_\varepsilon D - D \sum_{i=0}^N \beta_i (D^{-1}M)^i$ . However, given MDBD  $\mathcal{B}_{N,T}(\alpha, p)$  that induces a discretized p.d.f.  $\gamma$ , to apply the algorithm in [4] one has to explicitly precompute in  $O(NT)$  time the vector  $\gamma$ . Instead, we give two algorithms (sequential and parallel) that bypass this explicit precomputation.

We propose a faster sequential algorithm that on input MDBD  $\mathcal{B}_{N,T}(\alpha, p)$  with  $N = 2^k$  for  $k \in \mathbb{N}_+$  outputs in  $\hat{O}(\varepsilon^{-2}m + \varepsilon^{-4}nT)$  time the desired spectral sparsifier. Moreover, our algorithm is parallelizable and runs in  $\hat{O}(\varepsilon^{-2}m + \varepsilon^{-4}nT)$  work and  $O(\log N \cdot \text{poly}(\log n) + \log T)$  depth. Our main algorithmic contribution is to propose the first efficient parallel algorithm that on input continuous p.d.f.  $w \in \mathcal{C}$ , matrix  $B = D - M$  as above, outputs a spectral sparsifier of matrix-polynomial whose coefficients approximate component-wise the discretized p.d.f.  $\hat{w}$ .

Our results yield the first efficient and parallel algorithm that runs in nearly linear work and poly-logarithmic depth and analyzes the long term behaviour of Markov chains in non-trivial settings. In addition, we strengthen the Spielman and Peng’s [21] parallel SDD solver by introducing a simple parallel preprocessing step.

\*This work has been funded by the Cluster of Excellence “Multimodal Computing and Interaction” within the Excellence Initiative of the German Federal Government.

<sup>1</sup> $\hat{O}(\cdot)$  notation hides  $\text{poly}(\log n, \log N)$  factors.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Our Results</b>	<b>2</b>
2.1	Representational Power of MDBD . . . . .	3
2.2	Spectral Sparsification of Matrix Polynomials induced by MDBD . . . . .	4
2.3	Analyzing the Long Term Behaviour of Markov Chains . . . . .	6
2.4	Faster SDDM Solver . . . . .	6
<b>3</b>	<b>Algorithmic Background on Spectral Sparsification</b>	<b>7</b>
3.1	Prior Algorithms . . . . .	7
3.2	Spectral Sparsification of $\mathcal{T}$ -Matrices . . . . .	8
<b>4</b>	<b>Core Iterative Algorithm</b>	<b>9</b>
4.1	Initialization . . . . .	9
4.2	Iterative Construction . . . . .	10
<b>5</b>	<b>Spectral Sparsification of Binomial <math>\mathcal{T}</math>-Matrix Polynomials</b>	<b>11</b>
<b>6</b>	<b>Spectral Sparsification of <math>\mathcal{T}</math>-Matrix Polynomials Induced by MDBD</b>	<b>11</b>
<b>7</b>	<b>Parallelization of Algorithm SS_MDBD</b>	<b>12</b>
<b>8</b>	<b>Faster SDDM Solver</b>	<b>14</b>
<b>9</b>	<b>Representational Power of MDBD</b>	<b>14</b>
<b>10</b>	<b>Approximating Discretized PDF</b>	<b>16</b>
<b>11</b>	<b>Efficient Parallel Solver for Transpose Bernstein-Vandermonde Systems</b>	<b>18</b>
<b>A</b>	<b>Generalized Escaping Probability</b>	<b>21</b>
<b>B</b>	<b>Spectral Sparsification of <math>\mathcal{T}</math>-Matrices</b>	<b>21</b>
B.1	Structural Result . . . . .	22
<b>C</b>	<b>Bernstein Basis Matrix</b>	<b>23</b>
<b>D</b>	<b>Approximating Two Canonical PDFs</b>	<b>23</b>
D.1	Uniform Distribution . . . . .	23
D.2	Exponential Families . . . . .	24
<b>E</b>	<b>Schur Complement</b>	<b>25</b>

# 1 Introduction

In their seminal work Spielman and Teng [26] introduced the notion of spectral sparsifiers and proposed the first nearly linear time algorithm for spectral sparsification. In consecutive work, Spielman and Srivastava [24] proved that spectral sparsifiers with  $O(\varepsilon^{-2}n \log n)$  edges exist and can be computed in  $\tilde{O}(m \log^c n \cdot \log(w_{\max}/w_{\min}))^2$  time for any undirected graph  $G = (V, E, w)$ . The computational bottleneck of their algorithm is to approximate the solutions of logarithmically many SDD<sup>3</sup> systems.

Recently, Koutis, Miller and Peng [15] developed an improved solver for SDD systems that works in  $\tilde{O}(m \log n \cdot \log(1/\varepsilon))$  time. In a survey result [13, Theorem 3] Kelner and Levin showed that in  $\tilde{O}(m \log^2 n)$  time all effective resistances can be approximated up to a constant factor. This yields a  $(1 \pm \varepsilon)$ -spectral sparsifier with only a constant factor blow-up of non-zero edges  $O(\varepsilon^{-2}n \log n)$ . Although there are faster by a poly log-factor sparsification algorithms [16] they output spectral sparsifiers with poly log-factor more edges.

Spielman and Peng [21] introduced the notion of *sparse approximate inverse chain* of SDDM<sup>4</sup> matrices. They proposed the first parallel algorithm that finds such chains and runs in work  $\tilde{O}(m \log^3 n \cdot \log^2 \kappa)$  and depth  $O(\log^c n \cdot \log \kappa)$ , where  $\kappa$  is the condition number of the SDDM matrix with  $m$  non-zero entries and dimension  $n$ . Furthermore, they showed that in  $\tilde{O}(\varepsilon^{-2}m \log^3 n)$  time a spectral sparsifier  $\tilde{D} - \tilde{A} \approx_\varepsilon D - AD^{-1}A$  can be computed with  $nnz(\tilde{A}) \leq O(\varepsilon^{-2}n \log n)$ . In a follow up work, Cheng et al. [3] designed an algorithm that computes a sparse approximate generalized chain  $\tilde{C}$  such that  $\tilde{C}\tilde{C}^T \approx_\varepsilon M^p$  for any SDDM matrix  $M$  and  $|p| \leq 1$ . The chain  $\tilde{C}$  is constructed iteratively and it involves a normalization step that produces a sparsifier  $D - \tilde{M}_{i+1} \approx_\varepsilon D - \tilde{M}_i D^{-1} \tilde{M}_i$  that is expressed in terms of the original diagonal matrix  $D$ , for all iterations  $i$ .

Sinclair and Jerrum [23] analyzed Markov chains with transition matrices  $W = [I + D^{-1}A]/2$ , corresponding to lazy random walks. They proved that these walks converge fast to stationary distribution, defined by  $\pi_u = d_u / (\sum_{u \in V} d_u)$ , after  $O(\phi_G^{-2} \log(\min_{u \in V} \pi_u^{-1}))^5$  steps. Andersen et al. [1] gave an efficient local clustering algorithm that relies on a lazy variation of PageRank, the transition matrix of which is defined by  $\sum_{t=0}^{\infty} \alpha(1 - \alpha)^t W^t$ , where  $\alpha > 0$  is a parameter. Their local algorithm uses a truncated (finite summation) version of the preceding transition matrix.

Recently, Cheng et al. [4] initiated the study of computing spectral sparsifiers  $D - \hat{A} \approx_\varepsilon D - D \sum_{i=1}^N \xi_i (D^{-1}A)^i$  of random walk Laplacian matrix polynomials, where  $\xi$  is a probability distribution over  $[1 : N]$ ,  $D - A$  is a Laplacian matrix and  $\sum_{i=1}^N \xi_i (D^{-1}A)^i$  is a random walk transition matrix. These matrix polynomials capture the long term behaviour of Markov chains. Moreover, a sparsifier of a matrix polynomial yields a multiplicative approximation of the expected generalized “escaping probability” [10, 14] of random walks. Cheng et al. [4] gave the first sequential algorithm that computes a spectral sparsifier of a random walk Laplacian matrix polynomial and runs in time  $\tilde{O}(\varepsilon^{-2} \cdot mN^2 \cdot \log^{c_1} n \cdot \log^{c_2} N)$  for some small constants  $c_1, c_2$ .

## 2 Our Results

The lazy random walk length  $N$  in the regime of interest in [1, 10, 23] is of order  $N = \Theta(\text{poly}(n))$ . The quadratic runtime dependance on  $N$  makes the algorithm in [4] prohibitively expensive for analysing the long term behaviour of Markov chains. In this paper, we overcome this issue for

<sup>2</sup>The  $\tilde{O}(\cdot)$  notation hides  $O(\text{poly}(\log \log n))$  factors.

<sup>3</sup>SDD is the class of symmetric and diagonally dominant matrices.

<sup>4</sup>SDDM is the class of positive definite SDD matrices with non-positive off-diagonal entries.

<sup>5</sup>Graph conductance  $\phi_G \triangleq \min_{\mu(S) \leq \mu(V)/2} \phi(S)$ , where  $\phi(S) \triangleq |w(S, \bar{S})|/\mu(S)$  and  $\mu(S) \triangleq \sum_{u \in S} d_u$ .

“large” class of probability distributions  $\gamma$  over  $[0 : N]$  that are induced by mixture of discrete Binomial distributions (MDBD) with  $N = 2^k$  for  $k \in \mathbb{N}_+$ . Our results are summarized as follows.

In Subsection 2.1, we analyze the representational power of MDBD. In Subsection 2.2, we give a sequential and a parallel algorithm for computing a spectral sparsifier of matrix polynomials induced by MDBD. In Subsection 2.3, we propose the first parallel algorithm that runs in nearly linear work and poly-logarithmic depth and analyzes the long term behaviour of Markov chains in non-trivial settings. In Subsection 2.4, we strengthen the Spielman and Peng’s [21] parallel SDD solver.

## 2.1 Representational Power of MDBD

Let  $B(p, N)$  be Binomial distribution for some parameters  $p \in (0, 1)$  and  $N \in \mathbb{N}_+$ . MDBD is a set of pairs  $\{(B(p_i, N), \alpha_i)\}_{i=1}^T$ , denoted by  $\mathcal{B}_{N,T}(\alpha, p)$ , that satisfies the following two conditions:

1. (distinctness)  $p_i \in (0, 1)$  and  $p_i \neq p_j$  for all  $i \neq j \in [1 : T]$ ;
2. (positive linear combination)  $\sum_{i=1}^T \alpha_i \leq 1$  and  $\alpha_i \in (0, 1)$  for all  $i \in [1 : T]$ .

We prove in Section 9 that for every function  $w$  in a “large” class of continuous p.d.f., there exists MDBD that induces a component-wise approximation of  $w$ .

**Theorem 2.1** (MDBD Yields a Component-Wise Approximation). *Let  $w(x)$  be a four times differentiable p.d.f.,  $\varepsilon_I > 0$  a parameter and  $I = [0, 1]$  an interval. Suppose there is an integer  $N_0 \in \mathbb{N}$  and reals  $\mu \in (0, 1)$  and  $\phi_w \geq 1$  such that:*

- 1)  $\max_{x \in I} |w''(x)| \leq 2\phi_w \cdot N_0^2$ ,    2)  $\max_{x \in I} |w'(x)| \leq \frac{1}{2}\phi_w \cdot N_0$ ,    3)  $\max_{x \in I} |w(x)| \leq \phi_w$ ,
  - 4)  $\max_{x \in I} |b_2(x)| \leq \frac{1}{2}\mu \cdot N_0^2$ ,    5)  $\max_{x \in I} |b_1(x)| \leq \frac{1}{2}\mu \cdot N_0$ ,
- where the functions  $b_1, b_2$  are defined by  $b_1(x) = \frac{1}{w(x)}[-w(x) + (1-2x)w'(x) + \frac{1}{2}x(1-x)w''(x)]$  and  $b_2(x) = \frac{1}{w(x)}[w(x) - 3(1-2x)w'(x) + (1-6x+6x^2)w''(x) + \frac{5}{6}x(1-x)(1-2x)w'''(x) + \frac{1}{8}x^2(1-x)^2w^{(4)}(x)]$ . Then for every  $N \geq N_0$ , any  $T \geq \Omega(N\sqrt{\phi_w/\varepsilon_I})$  and all  $i \in [3 : N-3]$  there is  $\eta_i \in [-\mu, \mu]$  such that

$$\left| (1 + \eta_i) \frac{w(i/N)}{N} - \sum_{j=1}^T F_i(j/[T+1]) \right| \leq \frac{\varepsilon_I}{N}, \quad (1)$$

where  $F_i(x) \triangleq (w(x)/[T+1]) \cdot B_{N,i}(x)$  and  $B_{N,i}(x) \triangleq \binom{N}{i} x^i (1-x)^{N-i}$ .

For every function  $w$  that satisfies the hypothesis in Theorem 2.1 we associate MDBD  $\mathcal{B}_{N,T}(\alpha, p)$  that is defined by  $p_j = j/(T+1)$  and  $\alpha_j = w(p_j)/(T+1)$  for all  $j \in [1 : T]$ . Moreover, in Section 10 we give an efficient parallel algorithm that on input a continuous p.d.f.  $w$  (satisfying the conditions in Theorem 2.2) and integer  $N \in \mathbb{N}_+$ , outputs MDBD that induces a discretized p.d.f. which approximates component-wise a desired discretized p.d.f.  $\hat{w}$ .

**Theorem 2.2** (An Efficient Parallel Algorithm for Finding MDBD). *Let  $w(x) = C \cdot f(x)$  be a p.d.f. that satisfies the hypothesis of Theorem 2.1. Suppose also a)  $0 \leq f(x) \leq 1$ , b)  $\frac{1}{2}[f(0) + f(1)] \geq \Omega(1)$ , c)  $1 \leq C \leq o(N)$ , and d)  $\int_0^1 |f^{(2)}(x)| dx \leq o(N)$ . Then there is a parallel algorithm **AppDscrPDF** that on input  $w(x)$  as above, integer  $N \in \mathbb{N}_+$  and parameter  $\varepsilon_I > 0$ , outputs in  $O(N\sqrt{\phi_w/\varepsilon_I})$  work and  $O(\log(N\sqrt{\phi_w/\varepsilon_I}))$  depth MDBD  $\mathcal{B}_{N,T}(\alpha, p)$  that induces a discretized probability distribution  $\gamma/[1-\delta_w]$  over  $[0 : N]$  such that for all  $i \in [3 : N-3]$*

$$\frac{\gamma_i}{1-\delta_w} \in \left[ (1 + 2\eta_i) \cdot \hat{w}(i/N) \pm \frac{2\varepsilon_I}{S_{N+1,N}} \right],$$

where the target discretized p.d.f. is defined by  $\hat{w}(i/N) \triangleq w(i/N)/S_{N+1,N}$  for all  $i \in [0 : N]$ ,  $\delta_w \triangleq 1 - \frac{S_{T,T+1}/(T+1)}{S_{N+1,N/N}}$ ,  $S_{N+1,N} \triangleq \sum_{j=0}^N w(j/N)$  and  $S_{T,T+1} \triangleq \sum_{k=1}^T w(j/[T+1])$ . Moreover, it holds that  $\delta_w \in [0, o(1)]$ .

In Appendix D, we illustrate the representational power of MDBD by applying Theorem 2.2 for two canonical continuous p.d.f.: the Uniform distribution and the Exponential Families.

**Exact Recovery** Interestingly, in case when a discretized p.d.f.  $\hat{w}$  is induced by MDBD  $\mathcal{B}_{N,T}(\alpha, p)$  with exactly  $T = N + 1$  distinct Binomial distributions, we give in Section 11 an efficient parallel algorithm that on input vectors  $p$  and  $\hat{w}$ , outputs the vector  $\alpha$  in  $O(N \log^2 N)$  work and  $O(\log^c N)$  depth for some constant  $c \in \mathbb{N}_+$ .

**Theorem 2.3** (Canonical Instances Admit Exact Recovery). *Suppose  $p \in (0, 1)^{N+1}$  is a vector such that  $0 < p_i \neq p_j < 1$  for every  $i \neq j$  and  $\hat{w} \in (0, 1)^{N+1}$  is a discretized p.d.f. that is induced by MDBD  $\mathcal{B}_{N,N+1}(\alpha, p)$  that satisfies  $\hat{w}(i) = \sum_{j=1}^{N+1} \alpha_j \cdot B_{N,i}(p_j)$  for every  $i \in [0 : N]$ . Then there is a parallel algorithm that on input the vectors  $p$  and  $\hat{w}$ , outputs the vector  $\alpha \in (0, 1)^{N+1}$  in  $O(N \log^2 N)$  work and  $O(\log^c n)$  depth, for some constant  $c \in \mathbb{N}_+$ .*

## 2.2 Spectral Sparsification of Matrix Polynomials induced by MDBD

A matrix  $B$  is  $\mathcal{T}$ -matrix if it is either Laplacian or SDDM matrix. To highlight that an algorithm  $\mathcal{A}$  preserves the matrix type, we write that the algorithm  $\mathcal{A}$  on input a  $\mathcal{T}$ -matrix  $B$  outputs a matrix  $B'$  that is also  $\mathcal{T}$ -matrix.

Moreover, we say that a matrix  $X$  is a spectral sparsifier of a matrix  $Y$  if it satisfies  $(1 - \varepsilon)Y \preceq X \preceq (1 + \varepsilon)Y$ , for short  $X \approx_\varepsilon Y$ , where the partial relation  $X \succeq 0$  stands for  $X$  is symmetric positive semi-definite (SPSD) matrix.

We denote by  $\text{nnz}(A)$  or  $m_A$  the number of non-zero entries of matrix  $A$ . When we write “ $B = D - M$  is  $\mathcal{T}$ -matrix” we assume that  $D$  is positive diagonal matrix and  $B \in \mathbb{R}^{n \times n}$ . All algorithms presented in this paper output spectral sparsifiers *with high probability*.

**Sequential Algorithms** Cheng et al. [5, Theorem 1.5] gave an algorithm that on input a Laplacian matrix  $L = D - A$ , even integer  $N \in \mathbb{N}_+$  and parameter  $\varepsilon > 0$ , outputs in time  $O(\varepsilon^{-2} m_L \log^3 n \cdot \log^2 N)$  a spectral sparsifier  $D - \hat{A} \approx_\varepsilon D - D(D^{-1}A)^N$  of a matrix-monomial such that  $\text{nnz}(\hat{A}) \leq O(\varepsilon^{-2} n \log n)$ . In Section 4, we give for  $N = 2^k$  and  $k \in \mathbb{N}_+$  a  $O(\log^2 N)$ -factor faster algorithm that computes a spectral sparsifier of  $\mathcal{T}$ -matrix monomials. Furthermore, for any  $\mathcal{T}$ -matrix  $D - M$  such that  $M$  is PSD matrix, we prove that the initial sparsification step dominates the algorithm’s runtime.

**Theorem 2.4** (Power Method for Monomials). *There is an algorithm **PwrSS** that on input  $\mathcal{T}$ -matrix  $B = D - M$ ,  $N = 2^k$  for  $k \in \mathbb{N}_+$  and  $\varepsilon \in (0, 1)$ , outputs a spectral sparsifier  $D - \widehat{M}_N \approx_\varepsilon D - D(D^{-1}M)^N$  that is  $\mathcal{T}$ -matrix with  $\text{nnz}(\widehat{M}_N) \leq O(\varepsilon^{-2} n \log n)$ . The algorithm runs in time*

$$\begin{cases} \tilde{O}(m_B \log^2 n + \varepsilon^{-4} \cdot n \log^4 n \cdot \log^5 N) & , \text{ if } M \text{ is PSD matrix;} \\ \tilde{O}(\varepsilon^{-2} m_B \log^3 n + \varepsilon^{-4} \cdot n \log^4 n \cdot \log^5 N) & , \text{ otherwise.} \end{cases}$$

Using Theorem 2.4, we give in Section 5 an algorithm that runs by  $\Theta(N^2)$ -factor faster than [4, Theorem 2] and computes a spectral sparsifier of a single Binomial  $\mathcal{T}$ -matrix polynomials of the form  $D - D \sum_{i=0}^N B_{N,i}(p) \cdot (D^{-1}M)^i = D - DW_p^N$ , where  $W_p = (1 - p)I + pD^{-1}M$  and  $p \in (0, 1)$ .

**Theorem 2.5** (Single Binomial Matrix Polynomials). *There is an algorithm **LazySS** that on input  $\mathcal{T}$ -matrix  $B = D - M$ , number  $N = 2^k$  for  $k \in \mathbb{N}_+$ , and parameters  $\varepsilon, p \in (0, 1)$ , outputs a spectral sparsifier*

$$D - \widehat{M}_{p,N} \approx_\varepsilon D - DW_p^N = D - D \sum_{i=0}^N B_{N,i}(p) \cdot (D^{-1}M)^i$$

that is  $\mathcal{T}$ -matrix with at most  $O(\varepsilon^{-2}n \log n)$  non-zero entries. The algorithm **LazySS** runs in time

$$\begin{cases} \widetilde{O}(m_B \log^2 n + \varepsilon^{-4} \cdot n \log^4 n \cdot \log^5 N) & , \text{ if } p \in (0, 1/2]; \\ \widetilde{O}(\varepsilon^{-2}m_B \log^3 n \cdot \log^2 N + \varepsilon^{-4} \cdot n \log^4 n \cdot \log^5 N) & , \text{ otherwise.} \end{cases}$$

In Section 6, we give our main sequential algorithm that builds upon Theorem 2.5 and computes a spectral sparsifier of  $\mathcal{T}$ -matrix polynomials induced by MDBD.

**Theorem 2.6** (Mixture of Binomial Matrix Polynomials). *There is an algorithm **SS\_MDBD** that on input  $\mathcal{T}$ -matrix  $B = D - M$ , integer  $N = 2^k$ ,  $k \in \mathbb{N}_+$ , MDBD  $\mathcal{B}_{N,T}(\alpha, p)$  with  $\delta = 1 - \sum_{i=1}^T \alpha_i$  and parameter  $\varepsilon \in (0, 1)$ , outputs a spectral sparsifier  $D - \widehat{M} \approx_\varepsilon D - D \sum_{i=0}^N (\gamma_i / [1 - \delta]) \cdot (D^{-1}M)^i$  that is  $\mathcal{T}$ -matrix with  $O(\varepsilon^{-2}n \log n)$  non-zero entries, where  $\gamma$  is a discretized p.d.f. that satisfies  $\gamma_i = \sum_{j=1}^T \alpha_j \cdot B_{N,i}(p_j)$  for all  $i$ . The algorithm runs in time*

$$\begin{cases} \widetilde{O}(m_B \log^2 n + \varepsilon^{-4} \cdot nT \cdot \log^4 n \cdot \log^5 N) & , \text{ if } M \text{ is SPSD matrix;} \\ \widetilde{O}(\varepsilon^{-2}m_B \log^3 n + \varepsilon^{-4} \cdot nT \cdot \log^4 n \cdot \log^5 N) & , \text{ otherwise.} \end{cases}$$

We motivate now the first conclusion of Theorem 2.6. When  $B = D - DW_p$  is a Laplacian matrix (of a graph with self-loops) associated with a Markov Chain with transition matrix  $W_p$  that corresponds to  $p$ -lazy random walk process, it holds for  $p \in (0, 1/2]$  (c.f. Lemma 5.1) that  $DW_p$  is SPSD matrix.

Given MDBD  $\mathcal{B}_{N,T}(\alpha, p)$  that induces a vector  $\gamma$ , the algorithm in [4, Theorem 2] outputs a spectral sparsifier of the corresponding  $\mathcal{T}$ -matrix polynomial in time  $\widehat{O}(\varepsilon^{-2}mN^2 + NT)^6$ , where the term  $O(NT)$  accounts for computing the vector  $\gamma$ . In comparison, our improved algorithm **SS\_MDBD** runs in time  $\widehat{O}(\varepsilon^{-2}m + \varepsilon^{-4}nT)$  for any MDBD with  $N = 2^k$  for  $k \in \mathbb{N}_+$ .

**Parallel Algorithms** Building upon the seminal works of Spielman and Teng [25], Orecchia and Vishnoi [20] and Spielman and Peng [21], we prove in Section 7 that algorithm **SS\_MDBD** can be efficiently parallelized. To the best of our knowledge, this is the first efficient and parallel algorithm that sparsifies  $\mathcal{T}$ -matrix polynomials induced by MDBD with  $N = 2^k$  for  $k \in \mathbb{N}_+$ .

**Theorem 2.7** (Efficient Parallel Spectral Sparsification of Matrix Polynomial induced by MDBD). *There is a parallel algorithm **pSS\_MDBD** that on input as in Theorem 2.6, outputs a spectral sparsifier  $D - \widehat{M} \approx_\varepsilon D - D \sum_{i=0}^N (\gamma_i / [1 - \delta]) \cdot (D^{-1}M)^i$  that is  $\mathcal{T}$ -matrix with  $nnz(\widehat{M}) \leq O(\varepsilon^{-2}n \log^c n)$  for some constant  $c$ , where  $\gamma$  is a discretized p.d.f. such that  $\gamma_i = \sum_{j=1}^T \alpha_j \cdot B_{N,i}(p_j)$  for all  $i$ . The algorithm runs in work  $\widetilde{O}(\varepsilon^{-2}m_B \log^{c_1+1} n \cdot \log^2 N + \varepsilon^{-4} \cdot nT \cdot \log^{c+c_1+1} n \cdot \log^5 N)$  and depth  $O(\log^{c_2} n \cdot \log N + \log T)$  for constants  $c_1, c_2 \in \mathbb{N}$ .*

By combining Theorem 2.2 and Theorem 2.7, we develop an efficient parallel algorithm that outputs a spectral sparsifier of a  $\mathcal{T}$ -matrix polynomial whose coefficients approximate component-wise a target discretized p.d.f.  $\widehat{w}$ .

---

<sup>6</sup> $\widehat{O}(\cdot)$  notation hides  $\text{poly}(\log n, \log N)$  factors.



**Corollary 2.8** (Approximating Target Transition Matrices). *There is a parallel algorithm that takes as input a continuous p.d.f.  $w$  satisfying the conditions of Theorem 2.2,  $\mathcal{T}$ -matrix  $B = D - M$  and parameters  $\varepsilon, \varepsilon_I \in (0, 1)$ , and it outputs in  $\tilde{O}(\varepsilon^{-2}m_B + \varepsilon^{-4}nN\sqrt{\phi_w/\varepsilon_I})$  work and  $O(\log^{c_2} n \cdot \log N + \log(N\sqrt{\phi_w/\varepsilon_I}))$  depth a spectral sparsifier  $D - \hat{M} \approx_\varepsilon D - D \sum_{i=0}^N (\gamma_i/[1 - \delta_w]) \cdot (D^{-1}M)^i$  that is  $\mathcal{T}$ -matrix with  $\text{nnz}(\hat{M}) \leq O(\varepsilon^{-2}n \log^c n)$  such that for all  $i \in [3 : N - 3]$  it holds  $\gamma_i/[1 - \delta_w] \in [(1 + 2\eta_i) \cdot \hat{w}(i/N) \pm 2\varepsilon_I/S_{N+1,N}]$ .*

### 2.3 Analyzing the Long Term Behaviour of Markov Chains

For many finite Markov chains [1, 10, 14, 23] there exists  $N' \in \mathbb{N}_+$  such that for every  $N \geq N'$  certain phenomenon occurs with high probability - (local) mixing time, truncated PageRank, etc. Therefore, to analyze the long term behaviour of a finite Markov chain, it suffices to select the smallest  $N = 2^k$  for  $k \in \mathbb{N}_+$  that is larger or equal to  $N'$ .

**Corollary 2.9** (Capturing The Long Term Behaviour of Markov Chains). *Suppose  $L = D - A$  is dense Laplacian matrix with  $m_L = \Theta(n^2)$ ,  $\varepsilon \in (0, 1)$ ,  $\mathcal{B}_{N,T}(\alpha, p)$  is MBD such that  $\sum_{i=1}^T \alpha_i = 1$ , the degree  $N = 2^k \leq O(n)$  for  $k \in \mathbb{N}_+$  and the number of Binomials  $T \leq O(n)$ . Then algorithm **pSS-MBD** outputs in work  $\tilde{O}(\varepsilon^{-4} \cdot m_L \cdot \log^{c+c_1+6} n)$  and depth  $O(\log^{c_2+1} n)$  a spectral sparsifier  $D - \hat{A} \approx_\varepsilon D - D \sum_{i=0}^N \gamma_i (D^{-1}A)^i$  that is Laplacian matrix with  $O(\varepsilon^{-2}n \log^{c_2} n)$  non-zero entries for some constants  $c, c_1, c_2 \in \mathbb{N}$  and  $\gamma$  is a probability distribution induced by the MBD  $\mathcal{B}_{N,T}(\alpha, p)$ .*

**Multiplicative Approximation of Generalized Escaping Probability** Consider a Markov chain with transition matrix  $\mathcal{G}_\gamma = \sum_{i=0}^N \gamma_i (D^{-1}A)^i$  that corresponds to a generalized random walk process of length  $N$ . Perform a random walk of length  $N$  induced by  $\mathcal{G}_\gamma$  that starts at vertex  $v \in V$ . Then for any subset  $S \subset V$ , the corresponding *generalized escaping probability* is defined by  $\text{gEsc}(v, S, \mathcal{G}_\gamma) = \mathbf{1}_v^T \mathcal{G}_\gamma \mathbf{1}_{\bar{S}}$ , where we denote by  $\mathbf{1}_T$  the characteristic vector of a subset  $T \subset V$ .

We define the volume of  $S$  by  $\mu(S) = \sum_{u \in S} d_u$  and let  $\pi_S$  be a probability distribution over  $V$  defined by  $\pi_S(u) = d_u/\mu(S)$  if  $u \in S$  and  $\pi_S(u) = 0$  otherwise. The *expected* generalized escaping probability (E.G.E.P.) with respect to  $\pi_S$  is defined by

$$\mathbb{E}_{v \sim \pi_S} [\text{gEsc}(v, S, \mathcal{G}_\gamma)] = \pi_S^T \mathcal{G}_\gamma \mathbf{1}_{\bar{S}}. \quad (2)$$

We show in Appendix A that a spectral sparsifier of a random walk Laplacian matrix polynomial, yields a multiplicative approximation of E.G.E.P. for all subsets  $S \subset V$ .

**Lemma 2.10** (Multiplicative Approximation of E.G.E.P.). *For any spectral sparsifier  $D - \hat{A}_N \approx_\varepsilon D - D \sum_{i=0}^N \gamma_i (D^{-1}A)^i$  of a random walk Laplacian matrix polynomial such that  $\gamma$  is a probability distribution over  $[0 : N]$ , it holds for every subset  $S \subset V$  that*

$$\xi_S^T (D - \hat{A}_N) \xi_S \in [(1 \pm \varepsilon) \cdot \mathbb{E}_{v \sim \pi_S} [\text{gEsc}(v, S, \mathcal{G}_\gamma)]], \quad \text{where} \quad \xi_S \triangleq \mathbf{1}_S / \sqrt{\mu(S)}.$$

Using Corollary 2.9 and Lemma 2.10, we propose the first efficient and parallel algorithm that runs in nearly linear work and poly-logarithmic depth that yields a multiplicative approximation of E.G.E.P. for Markov chains with transition matrices induced by MBD.

### 2.4 Faster SDDM Solver

Spielman and Peng [21] gave the first parallel SDD solver that constructs in  $O(m \log^{c_1} n \cdot \log^3 \kappa)$  work and  $O(\log^{c_2} n \cdot \log \kappa)$  depth a sparse  $O(1)$ -approximate inverse chain that solves approximately to any  $\varepsilon > 0$  precision an SDD system in  $O((m+n \log^c n \cdot \log^3 \kappa) \log 1/\varepsilon)$  work and  $O(\log n \cdot \log \kappa \cdot \log 1/\varepsilon)$  depth. In Section 8, we give a simple parallel preprocessing step that strengthens their algorithm.

**Theorem 2.11.** *There is an algorithm that on input an  $n$ -dimensional SDDM matrix  $M$  with  $m$  non-zeros and condition number at most  $\kappa$ , produces with probability at least  $1/2$  a sparse  $O(1)$ -approximate inverse chain that can be used to solve any linear equation in  $M$  to any precision  $\varepsilon > 0$  in  $O(n \log^c n \cdot \log^3 \kappa \cdot \log 1/\varepsilon)$  work and  $O(\log n \cdot \log \kappa \cdot \log 1/\varepsilon)$  depth, for some constant  $c$ . The algorithm runs in  $O(m \log^{c_1} n)$  work and  $O(\log^{c_2} n \cdot \log \kappa)$  depth for some other constants  $c_1, c_2$ .*

For the current state-of-the-art result on parallel SDD solvers we refer the reader to the work of Lee et al. [17].

### 3 Algorithmic Background on Spectral Sparsification

We write  $X \approx_{\varepsilon_1 \oplus \varepsilon_2} Y$  to indicate  $(1 - \varepsilon_1)(1 - \varepsilon_2)Y \preceq X \preceq (1 + \varepsilon_1)(1 + \varepsilon_2)Y$ . Our analysis uses the following five basic facts (c.f. [2, 26]).

**Fact 3.1.** *For positive semi-definite (PSD) matrices  $X, Y, W$  and  $Z$  it holds*

- a. *if  $Y \approx_\varepsilon Z$  then  $X + Y \approx_\varepsilon X + Z$ ;*
- b. *if  $X \approx_\varepsilon Y$  and  $W \approx_\varepsilon Z$  then  $X + W \approx_\varepsilon Y + Z$ ;*
- c. *if  $X \approx_{\varepsilon_1} Y$  and  $Y \approx_{\varepsilon_2} Z$  then  $X \approx_{\varepsilon_1 \oplus \varepsilon_2} Z$ ;*
- d. *if  $X$  and  $Y$  are invertible matrices such that  $X \approx_\varepsilon Y$  then  $X^{-1} \approx_{2\varepsilon} Y^{-1}$ ,  $\forall \varepsilon \in (0, \frac{1}{2})$ ;*
- e. *for any matrix  $V$  if  $X \approx_\varepsilon Y$  then  $V^T X V \approx_\varepsilon V^T Y V$ .*

#### 3.1 Prior Algorithms

Our algorithms for computing spectral sparsifiers of matrix-polynomials use as a black-box several spectral sparsification algorithms for Laplacian and SDDM matrices.

More precisely, our sequential algorithms build upon Theorem 3.2 that relies on Kelner and Levin's [13, Theorem 3] and Cohen et al.'s [6, Lemma 4], and Theorem 3.3 proposed by Spielman and Peng [21, Corollary 6.4].

**Theorem 3.2.** [13] *There is an algorithm **SS** takes as input parameter  $\varepsilon \in (0, 1)$ , matrices  $D$  and  $A$  such that  $D$  is positive diagonal and  $A$  is symmetric non-negative with  $A_{ii} = 0$  for all  $i$  such that  $L = D - A$  is Laplacian matrix. Then in  $\tilde{O}(m_L \log^2 n)$  time outputs a positive diagonal matrix  $\tilde{D}$  and symmetric non-negative matrix  $\tilde{A}$  such that  $\text{nnz}(\tilde{A}) \leq O(\varepsilon^{-2} n \log n)$ ,  $\tilde{A}_{ii} = 0$  for all  $i$ , and  $\tilde{D} - \tilde{A} \approx_\varepsilon D - A$ . Moreover,  $\tilde{D} - \tilde{A}$  is Laplacian matrix.*

**Theorem 3.3.** [21] *There is an algorithm **PS** that takes as input SDDM matrix  $B = D - M$  and parameter  $\varepsilon \in (0, 1)$ . Then in  $O(\varepsilon^{-2} m_B \log^2 n)$  time outputs a positive diagonal matrix  $\tilde{D}$  and symmetric non-negative matrix  $\tilde{M}$  with  $\text{nnz}(\tilde{M}) \leq O(\varepsilon^{-2} m_B \log n)$  and  $\tilde{M}_{ii} = 0$  for all  $i$ , such that  $\tilde{D} - \tilde{M} \approx_\varepsilon D - M D^{-1} M$  and  $\tilde{D} \approx_\varepsilon D$ . Moreover,  $\tilde{D} - \tilde{M}$  is SDDM matrix.*

Our parallel algorithm uses Theorem 3.4, which is the culmination of a research line conducted by Spielman and Teng [25], Orecchia and Vishnoi [20] and Spielman and Peng [21].

**Theorem 3.4.** [21] *There is an algorithm **STOVP** takes as input a Laplacian matrix  $D - M$  and parameter  $\varepsilon \in (0, 1/2)$ . Then it outputs a spectral sparsifier  $D - \tilde{M} \approx_\varepsilon D - M$  with  $\tilde{M}_{ii} = 0$  for all  $i$  and  $\text{nnz}(\tilde{M}) \leq O(\varepsilon^{-2} n \log^c n)$  for some constant  $c$ . Moreover, this algorithm requires  $O(m \log^{c_1} n)$  work and  $O(\log^{c_2} n)$  depth, for some other constants  $c_1$  and  $c_2$ .*

Based on Theorem 3.4 Spielman and Peng [21] parallelized algorithm **PS** (c.f. Theorem 3.3).



**Theorem 3.5.** [21] *There is a parallel algorithm that on input an SDDM matrix  $D - M$  and parameter  $\varepsilon \in (0, 1/2)$ , outputs a spectral sparsifier  $D - \widetilde{M} \approx_\varepsilon D - MD^{-1}M$  with  $\widetilde{D} \approx_\varepsilon D$ ,  $\widetilde{M}_{ii} = 0$  for all  $i$  and  $\text{nnz}(\widetilde{M}) \leq O(\varepsilon^{-2}n \log^c n)$  for some constant  $c$ . Moreover, this algorithm requires  $O(\varepsilon^{-2}m \log^{c_1+1} n)$  work and  $O(\log^{c_2} n)$  depth, for some other constants  $c_1$  and  $c_2$ .*

### 3.2 Spectral Sparsification of $\mathcal{T}$ -Matrices

We show that the algorithms **SS** and **PS** can be amended to produce  $\mathcal{T}$ -matrix sparsifiers that are in *normalized form*, i.e. the sparsifiers are expressed in terms of the diagonal matrix  $D$  minus a symmetric non-negative matrix  $\widehat{M}$ . Our analysis relies on several results established by Peng et al. [3, 4, 21, 22].

**Lemma 3.6.** *There is an algorithm **mSS** that takes as input a positive diagonal matrix  $D$ , symmetric non-negative matrix  $A$  (possibly  $A_{ii} \neq 0$ ) such that  $B = D - A$  is Laplacian matrix and parameter  $\varepsilon \in (0, 1)$ . Then it outputs in  $\tilde{O}(m_B \log^2 n)$  time a spectral sparsifier  $D - \widehat{A} \approx_\varepsilon D - A$  that is Laplacian matrix and satisfies  $\widehat{A}$  is symmetric non-negative matrix with  $\text{nnz}(\widehat{A}) \leq O(\varepsilon^{-2}n \log n)$ .*

The next result implicitly appears in [4]. For completeness we prove it in Appendix B.

**Lemma 3.7.** *Suppose  $D - A$  is Laplacian matrix (possibly  $A_{ii} \neq 0$ ) and  $\widetilde{D} - \widetilde{A}$  a sparsifier with  $\widetilde{A}_{ii} = 0$  for every  $i$  such that  $(1 - \varepsilon)(D - A) \preceq \widetilde{D} - \widetilde{A} \preceq (1 + \varepsilon)(D - A)$ . Then the symmetric non-negative matrix  $\widehat{A} = (D - \frac{1}{1+\varepsilon}\widetilde{D}) + \frac{1}{1+\varepsilon}\widetilde{A}$  satisfies  $(1 - 2\varepsilon)(D - A) \preceq D - \widehat{A} \preceq (1 + 2\varepsilon)(D - A)$ .*

We present now the proof of Lemma 3.6.

*Proof of Lemma 3.6.* Notice that  $D - A = D' - A'$ , where  $D'$  is positive diagonal matrix and  $A'$  is symmetric non-negative matrix such that  $A'_{ii} = 0$  for all  $i$ . By Theorem 3.2 we obtain a sparsifier  $\widetilde{D}' - \widetilde{A}' \approx_{\varepsilon/2} D' - A'$ . Then by Lemma 3.7 we have  $D' - \widehat{A}' \approx_\varepsilon D' - A'$ , where  $\widehat{A}' = (D' - \frac{1}{1+\varepsilon}\widetilde{D}') + \frac{1}{1+\varepsilon}\widetilde{A}'$  is symmetric non-negative matrix. We define by  $D_A = D - D'$  a non-negative diagonal matrix. Set  $\widehat{A} = D_A + \widehat{A}'$  and observe that it is symmetric and non-negative matrix. Now the statement follows since  $D - \widehat{A} = D' - \widehat{A}'$ . ■

**$\mathcal{T}$ -Matrices** Building upon the work of Spielman and Peng [21, Proposition 5.6] and Cheng et al. [4, Proposition 25], we prove in Appendix B the following statement.

**Lemma 3.8** (Closure). *Suppose  $D - M$  is  $\mathcal{T}$ -matrix. Then  $D - D(D^{-1}M)^N$  is  $\mathcal{T}$ -matrix for every  $N \in \mathbb{N}_+$ . Moreover, if  $D - \widehat{M} \approx_\varepsilon D - M$  is a spectral sparsifier, then  $D - D(D^{-1}\widehat{M})^N$  is  $\mathcal{T}$ -matrix for every  $N \in \mathbb{N}_+$ .*

**Normalized Algorithms** We present now two algorithms that sparsify matrices of the form  $D - D(D^{-1}M)^N$  for  $N \in \{1, 2\}$  such that the resulting sparsifiers are in normalized form.

**Lemma 3.9** (Normalized Spectral Sparsification). *There is an algorithm **mKLC** that takes as input  $\mathcal{T}$ -matrix  $B = D - M$  and parameter  $\varepsilon \in (0, 1)$ , then it outputs in  $\tilde{O}(m_B \log^2 n)$  time a spectral sparsifier  $D - \widehat{M} \approx_\varepsilon D - M$  that is  $\mathcal{T}$ -matrix and  $\widehat{M}$  is symmetric non-negative matrix with  $\text{nnz}(\widehat{M}) \leq O(\varepsilon^{-2}n \log n)$ .*

*Proof.* By definition  $B = D_1 + L$  where  $D_1$  is non-negative diagonal matrix and  $L = D_2 - M$  is Laplacian matrix. We obtain by Lemma 3.6 a sparsifier  $D_2 - \widehat{M} \approx_\varepsilon D_2 - M$  that is Laplacian matrix. Now we consider two cases. If  $D_1 = 0$  then we are done. Otherwise  $D_1$  is PSD matrix and by Fact 3.1.a we have  $D - \widehat{M} \approx_\varepsilon D - M$ . Since  $D - M$  is SDDM matrix and the operator  $\approx_\varepsilon$  preserves the kernel space, it follows that  $D - \widehat{M}$  is SDDM matrix. ■

We proceed by stating an interesting structural result that implicitly appears in [21] (c.f. Section “Efficient Parallel Construction”). For completeness we prove it in Appendix B.1.

**Lemma 3.10.** *Suppose  $B = D - M$  is  $\mathcal{T}$ -matrix. Let  $\eta_i = M_{i,:}^T - M_{i,i} \cdot \mathbf{1}_i$  be a column vector,  $d_i = \langle M_{i,:}, \mathbf{1} \rangle$  and  $s_i = d_i - M_{ii}$  numbers, and  $\mathbf{D}_{N_i} = (s_i/d_i) \cdot \text{diag}(N_i)$  positive diagonal matrix for all  $i$ , where  $N_i = \{M_{ij} \mid M_{ij} \neq 0\}$ . Let  $\mathbf{B}_{ij} = (M_{ii}/d_i + M_{jj}/d_j) \cdot M_{ij}$  be the  $(i, j)$ th entry of a matrix with same dimensions as matrix  $M$  and  $\mathbf{D}_B = \text{diag}(\mathbf{B} \cdot \mathbf{1})$  be a diagonal matrix.*

*Then it holds that  $D - MD^{-1}M = \mathbf{D}_1 + \mathbf{L}_B + \sum_{i=1}^n \mathbf{L}_{N_i}$  where  $\mathbf{D}_1 = \text{diag}[(D - MD^{-1}M)\mathbf{1}]$  is non-negative diagonal matrix,  $\mathbf{L}_B = \mathbf{D}_B - \mathbf{B}$  is Laplacian matrix with at most  $m_B$  non-zero entries and every  $\mathbf{L}_{N_i} = (s_i/d_i)\mathbf{D}_{N_i} - \eta_i\eta_i^T/d_i$  is Laplacian matrix corresponding to a clique with positively weighted edges that is induced by the neighbour set  $N_i$ .*

Spielman and Peng [21] gave algorithm **PS** (c.f. Theorem 3.3) for sparsifying matrices of the form  $D - MD^{-1}M$ , where  $D - M$  is SDDM matrix. We extend their result to  $\mathcal{T}$ -matrices and our algorithm outputs a spectral sparsifier in normalized form.

**Lemma 3.11** (Normalized 2-Hops Spectral Sparsification). *There is an algorithm **mPS** that on input a  $\mathcal{T}$ -matrix  $B = D - M$  and parameter  $\varepsilon \in (0, 1)$ , outputs in  $\tilde{O}(\varepsilon^{-2}m_B \log^3 n)$  time a spectral sparsifier  $D - \widehat{M} \approx_\varepsilon D - MD^{-1}M$  that is  $\mathcal{T}$ -matrix and  $\widehat{M}$  is symmetric non-negative matrix with  $\text{nnz}(\widehat{M}) \leq O(\varepsilon^{-2}n \log n)$ .*

*Proof.* By Lemma 3.6 we have  $D - MD^{-1}M = \mathbf{D}_1 + \mathbf{L}$ , where  $\mathbf{D}_1$  is non-negative diagonal matrix and  $\mathbf{L}$  is sum of Laplacian matrices. Using similar arguments as in “Section 6 Efficient Parallel Construction” [21] we find a sparsifier  $\widehat{D} - \widehat{M} \approx_{\varepsilon/2} \mathbf{L}$ . Moreover, we can compute the positive diagonal matrix  $D' = \text{diag}(\mathbf{L})$  in  $O(m_B)$  time (c.f. Appendix B.1), and then by Lemma 3.7 we obtain a sparsifier  $D' - \widehat{M} \approx_\varepsilon \mathbf{L}$ . Since  $\mathbf{D}_1$  is PSD matrix the statement follows by Fact 3.1.a. ■

## 4 Core Iterative Algorithm

Our goal now is to prove Theorem 2.4. We argue in a similar manner as in [5], but in contrast our analysis shows that the initial sparsification step tolerates higher approximation error. This observation yields an improved algorithm whose runtime is faster by a  $O(\log^2 N)$ -factor.

Moreover, we prove that for any  $\mathcal{T}$ -matrix  $D - M$  such that  $M$  is SPSD matrix, one can construct a spectral sparsifier  $D - \widehat{M}_2 \approx D - MD^{-1}M$  by first computing  $D - \widehat{M} \approx D - M$  and then  $D - \widehat{M}_2 \approx D - \widehat{M}D^{-1}\widehat{M}$ . This demonstrates that when  $M$  is SPSD matrix, the runtime is dominated by the initial sparsification.

The rest of this section is organized as follows. In Subsection 4.1 we describe the initial phase of algorithm **PwrSS**. Then in Subsection 4.2, we present the iterative construction of the desired spectral sparsifier  $D - \widehat{M}_N \approx_\varepsilon D - D(D^{-1}M)^N$ .

### 4.1 Initialization

We begin by extending [5, Lemma 4.3 and 4.4]. For completeness, we provide a prove in Appendix E where in addition we generalize [5, Fact 4.2].

**Lemma 4.1.** *Suppose  $B = D - M$  is  $\mathcal{T}$ -matrix and  $D - \widehat{M} \approx_\varepsilon D - M$  is a spectral sparsifier. If  $M$  is SPSD matrix then it holds that  $D - \widehat{M}D^{-1}\widehat{M} \approx_\varepsilon D - MD^{-1}M$ .*

Based on Lemma 4.1, we give a faster sparsification algorithm for  $\mathcal{T}$ -matrices  $D - MD^{-1}M$  such that  $M$  is SPSD matrix.

**Lemma 4.2.** *There is an algorithm **fSS** that takes as input  $\mathcal{T}$ -matrix  $B = D - M$  such that  $M$  is SPSP matrix, and parameter  $\varepsilon \in (0, 1)$ . Then it outputs in  $\tilde{O}(m_B \log^2 n + \varepsilon^{-4} n \log^4 n)$  time a spectral sparsifier  $D - \widehat{M}_2 \approx_{\varepsilon} D - MD^{-1}M$  that is  $\mathcal{T}$ -matrix with  $\text{nnz}(\widehat{M}_2) \leq O(\varepsilon^{-2} n \log n)$ .*

*Proof of Lemma 4.2.* We apply Lemma 3.9 to obtain a sparsifier  $D - \widehat{M} \approx_{\varepsilon/4} D - M$  in  $\tilde{O}(m_B \log^2 n)$  time with  $\text{nnz}(\widehat{M}) \leq O(\varepsilon^{-2} n \log n)$  such that  $D - \widehat{M}$  is  $\mathcal{T}$ -matrix. Then by Lemma 4.1 we know that  $D - \widehat{M}D^{-1}\widehat{M} \approx_{\varepsilon/4} D - MD^{-1}M$ . Now, by Lemma 3.8  $D - \widehat{M}D^{-1}\widehat{M}$  is  $\mathcal{T}$ -matrix. Then we apply Lemma 3.11 to obtain in  $\tilde{O}(\varepsilon^{-4} n \log^4 n)$  time a sparsifier  $D - \widehat{M}_2 \approx_{\varepsilon/4} D - \widehat{M}D^{-1}\widehat{M}$  with  $\text{nnz}(\widehat{M}_2) \leq O(\varepsilon^{-2} n \log n)$  such that  $D - \widehat{M}_2$  is  $\mathcal{T}$ -matrix. The claims follows by Fact 3.1.c. ■

## 4.2 Iterative Construction

Our analysis of the incurred approximation error after  $O(\log N)$  consecutive square sparsification operations builds upon [5, Lemma 4.1]. In contrast, we prove that for the initial and the final sparsifiers it suffices to have only an  $\varepsilon$  approximation, while all intermediate spectral sparsifiers require finer  $\varepsilon' = \Omega(\varepsilon / \log N)$  approximation. Due to this higher initial error tolerance, we improve the runtime of their algorithm by a  $O(\log^2 N)$ -factor.

**Lemma 4.3** (Accumulative Error). *Let  $D - M$  and  $D - \widehat{M}_2$  be  $\mathcal{T}$ -matrices such that  $D - \widehat{M}_2 \approx_{\varepsilon} D - MD^{-1}M$  and  $\text{nnz}(\widehat{M}_2) \leq O(\varepsilon^{-2} n \log n)$ . There is an algorithm **IndSS** that on input  $\mathcal{T}$ -matrix  $D - \widehat{M}_2$ , integer  $N = 2^k$  for  $k \in \mathbb{N}_+$  and parameter  $0 < \varepsilon' \leq \varepsilon$ , outputs in time  $\tilde{O}(\varepsilon'^{-4} n \log^4 n \cdot \log N)$  a symmetric non-negative matrix  $\widehat{M}_N$  with  $\text{nnz}(\widehat{M}_N) \leq O(\varepsilon^{-2} n \log n)$  such that  $D - \widehat{M}_N \approx_{(\oplus(\log N - 1)\varepsilon') \oplus 2\varepsilon} D - D(D^{-1}M)^N$  is  $\mathcal{T}$ -matrix.*

Our goal now is to prove Lemma 4.3. We establish next a useful algebraic property that all matrices of the form  $D(D^{-1}M)^{2^k}$  have in common.

**Lemma 4.4.** *If  $M$  is symmetric matrix, then  $D(D^{-1}M)^{2^k}$  is SPSP matrix for every  $k \in \mathbb{N}_+$ .*

*Proof.* Let  $Y \triangleq D^{-1/2}MD^{-1/2}$ . Notice that  $D(D^{-1}M)^{2^k} = D^{1/2}Y^{2^k}D^{1/2} = X^T X$ , where  $X = Y^{2^{k-1}}D^{1/2}$ . The statement follows since  $X^T X$  is SPSP matrix. ■

We present now the main iterative procedure used in algorithm **IndSS**.

**Lemma 4.5** (Iterative Procedure). *Let  $D - M$  and  $D - \widehat{M}_{2^k}$  be  $\mathcal{T}$ -matrices such that  $D - \widehat{M}_{2^k} \approx_{\varepsilon} D - D(D^{-1}M)^{2^k}$ , for  $k \in \mathbb{N}_+$ . There is an algorithm **SqrSS** that takes as input the  $\mathcal{T}$ -matrix  $D - \widehat{M}_{2^k}$  and parameter  $\varepsilon' \in (0, 1)$ , then it outputs in  $\tilde{O}(\varepsilon'^{-2} \text{nnz}(\widehat{M}_{2^k}) \log^3 n)$  time a symmetric non-negative matrix  $\widehat{M}_{2^{k+1}}$  with  $\text{nnz}(\widehat{M}_{2^{k+1}}) \leq O(\varepsilon'^{-2} n \log n)$  such that  $D - \widehat{M}_{2^{k+1}} \approx_{\varepsilon \oplus \varepsilon'} D - D(D^{-1}M)^{2^{k+1}}$  is  $\mathcal{T}$ -matrix.*

*Proof.* By Lemma 4.4,  $D(D^{-1}M)^{2^k}$  is SPSP matrix for any  $k \in \mathbb{N}_+$ . By Lemma 3.8 both  $D - D(D^{-1}M)^{2^k}$  and  $D - \widehat{M}_{2^k}D^{-1}\widehat{M}_{2^k}$  are  $\mathcal{T}$ -matrices. Hence, by Lemma 4.1 we have that  $D - \widehat{M}_{2^k}D^{-1}\widehat{M}_{2^k} \approx_{\varepsilon} D - D(D^{-1}M)^{2^{k+1}}$ . Now by Lemma 3.11 we have  $D - \widehat{M}_{2^{k+1}} \approx_{\varepsilon'} D - \widehat{M}_{2^k}D^{-1}\widehat{M}_{2^k}$  and hence the statement follows by Fact 3.1.c. ■

Based on the preceding results we are ready to prove Lemma 4.3.

*Proof of Lemma 4.3.* By Theorem 3.3 in time  $\tilde{O}((\varepsilon' \cdot \varepsilon)^{-2} n \log^4 n)$  we can compute a spectral sparsifier  $D - \widehat{M}_4 \approx_{\varepsilon \oplus \varepsilon'} D - D(D^{-1}M)^4$  with  $\text{nnz}(\widehat{M}_4) \leq O(\varepsilon'^{-2} n \log n)$ . Then we apply  $(\log N - 1)$  times Lemma 4.5 to obtain in  $\tilde{O}(\varepsilon'^{-4} n \log^4 n \cdot \log N)$  time a spectral sparsifier  $D - \widehat{M}_N \approx_{(\oplus(\log N - 1)\varepsilon') \oplus \varepsilon} D - D(D^{-1}M)^N$  with  $\text{nnz}(\widehat{M}_N) \leq O(\varepsilon'^{-2} n \log n)$ . The statement follows by applying Theorem 3.2 with  $\varepsilon$  to compute a refined spectral sparsifier of  $D - \widehat{M}_N$ . ■

**Proof of Theorem 2.4** In the initial phase we compute a sparsifier  $D - \widehat{M}_2 \approx_{\varepsilon/4} D - MD^{-1}M$  with  $nnz(\widehat{M}_2) \leq O(\varepsilon^{-2}n \log n)$  using either Lemma 4.2 (when  $M$  is SPSP matrix) or Lemma 3.11. The statement follows by applying Lemma 4.3 with  $\varepsilon' = \Omega(\varepsilon/\log N)$  to the sparsifier  $D - \widehat{M}_2$ .

## 5 Spectral Sparsification of Binomial $\mathcal{T}$ -Matrix Polynomials

In this section, we prove Theorem 2.5. We analyze first the properties of matrices of the form  $W_p = (1-p)I + pD^{-1}M$  for  $p \in (0, 1)$ . It is convenient to associate with them matrix-polynomials  $f_p(x) = (1-p) + px$  such that  $f_p(D^{-1}M) = W_p$ . Since the coefficients of the matrix-polynomial  $[f_p(x)]^N$  follow Binomial distribution  $B(N, p)$ , it follows that  $W_p^N = \sum_{i=0}^N B_{N,i}(p) \cdot (D^{-1}M)^i$  for every  $p \in (0, 1)$ , where  $B_{N,i}(p) = \binom{N}{i} p^i (1-p)^{N-i}$ .

When  $D - M$  is a Laplacian matrix, the matrix  $W_p^N$  corresponds to the transition matrix of a  $p$ -lazy random walk process of length  $N$  (c.f. [23]). We associate to such a Markov chain a matrix-polynomial  $D - DW_p^N = D - D \sum_{i=0}^N B_{N,i}(p) \cdot (D^{-1}M)^i$ . We present now some useful algebraic properties of matrices of the form  $DW_p^{2^k}$  and  $D - DW_p^N$ .

**Lemma 5.1.** *Suppose  $D - M$  is  $\mathcal{T}$ -matrix. Then  $D - DW_p^N$  is  $\mathcal{T}$ -matrix for every  $N \in \mathbb{N}_+$ . Also  $DW_p$  is SPSP matrix  $\forall p \in (0, 1/2]$  and  $DW_p^{2^k}$  is SPSP matrix  $\forall p \in (0, 1)$  and  $\forall k \in \mathbb{N}_+$ .*

*Proof.* By definition of  $W_p$ , we have  $D - DW_p = p(D - M)$  is  $\mathcal{T}$ -matrix. Suppose  $D - M$  is Laplacian matrix, then  $D - DW_p$  is Laplacian matrix and by Lemma 3.8,  $D - DW_p^N$  is Laplacian matrix for every  $N \in \mathbb{N}_+$ . Suppose now that  $D - M$  is SDDM matrix, then  $D - DW_p$  is SDDM matrix and by Lemma B.2  $D - DW_p^N$  is SDDM matrix for every  $N \in \mathbb{N}_+$ .

By definition  $DW_p = (1-p)D + pM$  and since  $D - M$  is diagonally dominant, it holds that  $DW_p$  is SPSP matrix for every  $p \in (0, 1/2]$ . Moreover, since  $DW_p$  is symmetric matrix by Lemma 4.4 it holds that  $D(D^{-1} \cdot DW_p)^{2^k} = DW_p^{2^k}$  is SPSP matrix for every  $k \in \mathbb{N}_+$ . ■

*Proof of Theorem 2.5.* The statement follows by Lemma 5.1 and Theorem 2.4. ■

## 6 Spectral Sparsification of $\mathcal{T}$ -Matrix Polynomials Induced by MDBD

Here we prove Theorem 2.6. Our approach relies on the following key algorithmic idea.

**Lemma 6.1** (Preprocessing of 2-Hop Spectral Sparsification). *Let  $D - M$  be a  $\mathcal{T}$ -matrix,  $D - \widehat{M}_1 \approx_\varepsilon D - M$  and  $D - \widehat{M}_2 \approx_\varepsilon D - MD^{-1}M$  are spectral sparsifiers. Then for every  $p \in (0, 1)$  the sparse matrix  $\widehat{M}_{p,2} = (1-p)^2 D + 2(1-p)p\widehat{M}_1 + p^2\widehat{M}_2$  yields a spectral sparsifier  $D - \widehat{M}_{p,2} \approx_\varepsilon D - DW_p^2$  that is  $\mathcal{T}$ -matrix.*

*Proof.* The statement follows by

$$\begin{aligned} D - \widehat{M}_{p,2} &= 2p(1-p)[D - \widehat{M}_1] + p^2[D - \widehat{M}_2] \\ &\approx_\varepsilon 2p(1-p)[D - M] + p^2[D - MD^{-1}M] = D - DW_p^2. \end{aligned}$$

■

Our algorithm **SS\_MDBD** builds upon Lemma 6.1 and Lemma 4.3. Due to the preprocessing step in Lemma 6.1 we speed up the sparsification of each  $\mathcal{T}$ -matrix polynomial  $D - DW_{p_j}^N$  for all  $j \in [1 : T]$ . We present now the pseudo code of algorithm **SS\_MDBD**.

---

**Algorithm 1**

---

- $(D, \widehat{M}) = \mathbf{SS\_MDBD}(D, M, \mathcal{B}_{N,T}(\alpha, p), \varepsilon)$
1. Let  $\varepsilon' = \varepsilon / \lceil 3 \log N \rceil$ ,  $\widehat{M}_{tmp} = 0$  and  $\delta = 1 - \sum_{i=1}^T \alpha_i$ .
  2.  $(D, \widehat{M}_1, \widehat{M}_2) = \mathbf{InitSS}(D, M, \varepsilon/3)$ .
  3. For every  $j \in \{1, \dots, T\}$  do
    - 3.1 Set  $p = p_{j,T} = j/(T+1)$  and  $\widehat{M}_{p,2} = (1-p)^2 D + 2p(1-p)\widehat{M}_1 + p^2 \widehat{M}_2$ .
    - 3.2  $(D, \widehat{M}_{p,N}) = \mathbf{IndSS}(\widehat{M}_{p,2}, N, \varepsilon')$  where  $D - \widehat{M}_{p,N} \approx_{2\varepsilon/3} D - DW_{p,N}^N$  (c.f. Lemma 4.3).
    - 3.3  $\widehat{M}_{tmp} = \widehat{M}_{tmp} + \alpha_j \cdot \widehat{M}_{p,N}$ .
  4. Sparsify  $D - \widehat{M} \approx_{\varepsilon/3} D - \frac{1}{1-\delta} \widehat{M}_{tmp}$  by algorithm **mKLC** (c.f. Lemma 3.9).
  5. Return  $(D, \widehat{M})$ .
- 

---

**Algorithm 2**

---

- $(D, \widehat{M}_1, \widehat{M}_2) = \mathbf{InitSS}(D, M, \varepsilon)$
1. Sparsify  $D - \widehat{M}_1 \approx_{\varepsilon} D - M$  by algorithm **mKLC** (c.f. Lemma 3.9).
  2. Sparsify  $D - \widehat{M}_2 \approx_{\varepsilon} D - MD^{-1}M$ 
    - 2.1 If  $M$  is SPSP matrix call algorithm **fSS** (c.f. Lemma 4.2),
    - 2.2 otherwise call algorithm **mPS** (c.f. Lemma 3.11).
  3. Return  $(D, \widehat{M}_1, \widehat{M}_2)$ .
- 

Let  $\delta = 1 - \sum_{i=1}^T \alpha_i$ . We denote a  $\mathcal{T}$ -matrix polynomial induced by MDBD  $\mathcal{B}_{N,T}(\alpha, p)$  as  $P_{\mathcal{B}} \triangleq \sum_{j=1}^T \alpha_j (D - DW_{p_j}^N) = (1 - \delta)D - D \sum_{i=0}^N \gamma_i (D^{-1}M)^i$ , where  $\gamma_i = \sum_{j=1}^T \alpha_j B_{N,i}(p_j)$ . We are now ready to prove Theorem 2.6.

*Proof of Theorem 2.6.* Let  $\varepsilon' = \varepsilon / \lceil 4 \log N \rceil$ . We perform first a preprocessing step. We apply Lemma 3.9 to obtain a sparsifier  $D - \widehat{M} \approx_{\varepsilon'} D - M$ . Then depending on whether  $M$  is SPSP matrix we use either Lemma 3.11 or Lemma 4.2 to obtain a sparsifier  $D - \widehat{M}_2 \approx_{\varepsilon'} D - MD^{-1}M$ . The run time is at most  $\tilde{O}(\varepsilon^{-2} m_B \log^3 n \cdot \log^2 N)$  or  $\tilde{O}(m_B \log^2 n + \varepsilon^{-4} n \log^4 n)$  respectively. Moreover, the sparsifiers satisfy  $\text{nnz}(\widehat{M}_1), \text{nnz}(\widehat{M}_2) \leq O(\varepsilon^{-2} n \log n \cdot \log^2 N)$ .

We combine Lemma 6.1 and Theorem 2.4 to find each sparsifier  $D - \widehat{M}_{p_j,N} \approx_{\varepsilon} D - DW_{p_j}^N$  by initializing algorithm **PwrSS** with a sparsifier  $D - \widehat{M}_{p_j,2} \approx_{\varepsilon'} D - DW_{p_j}^2$ . Let  $\widehat{M}_{tmp} = \sum_{j=1}^T \alpha_j \widehat{M}_{p_j,N}$ , then by Fact 3.1.b it holds  $(1 - \delta)D - \widehat{M}_{tmp} \approx_{\varepsilon} P_{\mathcal{B}}$ . This phase has  $\tilde{O}(\varepsilon^{-4} n T \log^4 n \cdot \log^5 N)$  runtime and each sparsifier satisfies  $\text{nnz}(\widehat{M}_{p_i,N}) \leq O(\varepsilon^{-2} n \log n)$ .

However, matrix  $\widehat{M}_{tmp}$  can be dense. We apply Lemma 3.9 to obtain a sparsifier  $D - \widehat{M} \approx_{\varepsilon} D - \frac{1}{1-\delta} \widehat{M}_{tmp} \approx_{\varepsilon} \frac{1}{1-\delta} P_{\mathcal{B}}$  in time  $\tilde{O}(\min\{n^2 \log^2 n, \varepsilon^{-2} n T \log^3 n \cdot \log^2 N\})$  such that  $\text{nnz}(\widehat{M}) \leq O(\varepsilon^{-2} n \log n)$ . ■

## 7 Parallelization of Algorithm SS\_MDBD

In this section we parallelize algorithm **SS\_MDBD**. This gives the first efficient parallel algorithm that computes a spectral sparsifier of any  $\mathcal{T}$ -matrix polynomial with coefficients induced by MDBD.

Our goal now is to prove Theorem 2.7. By construction of algorithms **SS\_MDBD** and **InitSS**, it suffices to show that we can efficiently parallelize algorithms **mKLC**, **mPS** and **PwrSS**. Then the statement follows by noting that each  $\mathcal{T}$  Binomial matrix-polynomial can be sparsified separately and in parallel.

We parallelize now algorithms **mKLC**, **mPS** and **IndSS**.

**Lemma 7.1.** *There is a parallel algorithm **pKLC** that on input  $\mathcal{T}$ -matrix  $B = D - M$  and parameter  $\varepsilon \in (0, 1)$ , outputs a spectral sparsifier  $D - \widehat{M} \approx_\varepsilon D - M$  that is  $\mathcal{T}$ -matrix such that  $\widehat{M}$  is symmetric non-negative matrix with  $\text{nnz}(\widehat{M}) \leq O(\varepsilon^{-2} n \log^c n)$  for some constant  $c$ . The algorithm runs in  $O(m_B \log^{c_1} n)$  work and  $O(\log^{c_2} n)$  depth, for some other constants  $c_1, c_2$ .*

*Proof.* We argue in a similar manner as in Lemma 3.9 to show that the statement holds for  $\mathcal{T}$ -matrices (Laplacian or SDDM matrices). Then the statement follows by Theorem 3.4. ■

**Lemma 7.2.** *There is a parallel algorithm **pPS** that on input  $\mathcal{T}$ -matrix  $B = D - M$  and parameter  $\varepsilon \in (0, 1/2)$ , outputs a spectral sparsifier  $D - \widetilde{M} \approx_\varepsilon D - MD^{-1}M$  that is  $\mathcal{T}$ -matrix such that  $\widetilde{D} \approx_\varepsilon D$ ,  $\widetilde{M}_{ii} = 0$  for all  $i$  and  $\text{nnz}(\widetilde{M}) \leq O(\varepsilon^{-2} n \log^c n)$  for some constant  $c$ . The algorithm runs in  $O(\varepsilon^{-2} m \log^{c_1+1} n)$  work and  $O(\log^{c_2} n)$  depth, for some other constants  $c_1$  and  $c_2$ .*

*Proof.* Using similar arguments as in Lemma 3.11 we prove that the statement holds for  $\mathcal{T}$ -matrices. Then the statement follows by Theorem 3.5. ■

**Lemma 7.3.** *Let  $D - M$  and  $D - \widehat{M}_{2^k}$  are  $\mathcal{T}$ -matrices such that  $D - \widehat{M}_{2^k} \approx_\varepsilon D - D(D^{-1}M)^{2^k}$ , for  $k \in \mathbb{N}_+$ . There is a parallel algorithm **pSqrSS** that on input the  $\mathcal{T}$ -matrix  $D - \widehat{M}_{2^k}$  and parameter  $\varepsilon' \in (0, 1)$ , outputs a spectral sparsifier  $D - \widehat{M}_{2^{k+1}} \approx_{\varepsilon \oplus \varepsilon'} D - D(D^{-1}M)^{2^{k+1}}$  that is  $\mathcal{T}$ -matrix with  $\text{nnz}(\widehat{M}_{2^{k+1}}) \leq O(\varepsilon'^{-2} n \log^c n)$  for some constant  $c$ . The algorithm runs in work  $O(\varepsilon'^{-2} \text{nnz}(\widehat{M}_{2^k}) \log^{c_1+1} n)$  and depth  $O(\log^{c_2} n)$ , for some other constants  $c_1, c_2$ .*

*Proof.* We use similar arguments as in Lemma 4.5, but we substitute Lemma 3.11 with Lemma 7.2. ■

**Lemma 7.4.** *Let  $D - M$  and  $D - \widehat{M}_2$  are  $\mathcal{T}$ -matrices such that  $D - \widehat{M}_2 \approx_\varepsilon D - MD^{-1}M$  and  $\text{nnz}(\widehat{M}_2) \leq O(\varepsilon^{-2} n \log^c n)$  for some constant  $c$ . There is a parallel algorithm **pIndSS** that on input  $\mathcal{T}$ -matrix  $D - \widehat{M}_2$ , integer  $N = 2^k$  for  $k \in \mathbb{N}$  and parameter  $0 < \varepsilon' \leq \varepsilon$ , outputs a spectral sparsifier  $D - \widehat{M}_N \approx_{(\oplus(\log N - 1)\varepsilon') \oplus 2\varepsilon} D - D(D^{-1}M)^N$  that is  $\mathcal{T}$ -matrix and  $\text{nnz}(\widehat{M}_N) \leq O(\varepsilon^{-2} n \log^c n)$ . The algorithm runs in work  $\widetilde{O}(\varepsilon'^{-4} n \log^{c+c_1+1} n)$  and depth  $O(\log^{c_2} n \cdot \log N)$  for some constants  $c_1, c_2$ .*

*Proof.* We argue in a similar manner as in Lemma 4.3. By Lemma 7.1 we compute a sparsifier  $D - \widehat{M}_4 \approx_{\varepsilon' \oplus \varepsilon} D - D(D^{-1}M)^4$  with  $\text{nnz}(\widehat{M}_4) \leq O(\varepsilon'^{-2} n \log^c n)$  in work  $O((\varepsilon \cdot \varepsilon')^{-2} n \log^{c+c_1+1} n)$  and depth  $O(\log^{c_2} n)$ . Then we apply  $(\log N - 1)$  times Lemma 7.3 to obtain a spectral sparsifier  $D - \widehat{M}_N \approx_{(\oplus(\log N - 1)\varepsilon') \oplus \varepsilon} D - D(D^{-1}M)^N$  with  $\text{nnz}(\widehat{M}_N) \leq O(\varepsilon'^{-2} n \log^c n)$  in  $O(\varepsilon'^{-4} n \log^{c+c_1+1} n)$  work and  $O(\log^{c_2} \cdot \log N)$  depth. The statement follows by applying Lemma 7.1 with  $\varepsilon$  to compute a refined spectral sparsifier of  $D - \widehat{M}_N$ . ■

We present now the proof of Theorem 2.7 which yields the parallel algorithm **pSS\_MDBD**.

*Proof of Theorem 2.7.* We sketch first our parallel algorithm **pSS\_MDBD**. We parallelize algorithm **InitSS** based on algorithms **pKLC** and **pPS**. Then, we sparsify separately and in parallel each of the  $T$  distinct single Binomial  $\mathcal{T}$ -matrix polynomials by algorithm **pIndSS**. The resulting  $T$  sparsifiers are scaled and merged into a  $\mathcal{T}$ -matrix polynomial induced by MDBD. Since this matrix-polynomial might be dense, we sparsify it using algorithm **pKLC**.

The correctness of algorithm **pSS\_MDBD** follows by Theorem 2.6. We analyze now the work and the depth of algorithm **pSS\_MDBD**. By Lemma 7.1 and Lemma 7.2 the initial phase is dominated by  $O(\varepsilon^{-2} m \log^{c_1+1} n \cdot \log^2 N)$  work and  $O(\log^{c_2} n)$  depth. Moreover, each of the sparsifiers  $D - \widehat{M} \approx_\varepsilon D - M$  and  $D - \widehat{M}_2 \approx_\varepsilon D - MD^{-1}M$  has at most  $O(\varepsilon^{-2} n \log^c n)$  non-zero entries.



Let  $\varepsilon' = \varepsilon/[12 \log N]$ . By Lemma 7.4 for each  $j \in [1 : T]$  we apply algorithm **pIndSS** to compute a spectral sparsifier  $D - \widehat{M}_{p_j, N} \approx_{\varepsilon/6} D - DW_{p_j}^N$  with  $\text{nnz}(\widehat{M}_{p_j, N}) \leq O(\varepsilon^{-2} n \log^c n)$ . This phase runs in work  $\tilde{O}(\varepsilon'^{-4} n T \log^{c+c_1+1} n)$  and depth  $O(\log^{c_2} n \cdot \log N)$ . Furthermore, the linear combination  $\widehat{M}_{tmp} = \sum_{j=1}^T \alpha_j \cdot \widehat{M}_{p_j, N}$  (in Step 3.3) can be computed in depth  $O(\log T)$ .

Therefore, we approximate by  $D - \frac{1}{1-\delta} \widehat{M}_{tmp} \approx_{\varepsilon/6} D - D \sum_{i=0}^N \frac{1}{1-\delta} \gamma_i (D^{-1} M)^i$  the desired  $\mathcal{T}$ -matrix polynomial induced by MDBD. Since matrix  $\widehat{M}_{tmp}$  might be dense, by Lemma 7.1 we compute a spectral sparsifier  $D - \widehat{M} \approx_{\varepsilon/12} D - \frac{1}{1-\delta} \widehat{M}_{tmp}$  with  $\text{nnz}(\widehat{M}) \leq O(\varepsilon^{-2} n \log^{c_2} n)$ . Algorithm **pKLC** runs in work  $O(\min\{\varepsilon^{-2} n T \log^c n, n^2\} \cdot \log^{c_1} n)$  and depth  $O(\log^{c_2} n)$ . ■

## 8 Faster SDDM Solver

In this section we prove Theorem 2.11. We argue in a similar manner as in [21], but in contrast our improved analysis relies on the refined initialization phase developed in Section 4 and its consecutive parallelization in Section 7. Spielman and Peng's [21] proof involves two major steps: the first is to construct a sparse  $O(1)$ -approximate inverse chain, and the second is to apply this chain as a preconditioner into an algorithm known as "Preconditioned Richardson Iteration" [21, Lemma 4.4].

We give now an improved construction for a sparse  $\varepsilon$ -approximate inverse chain. This directly implies the desired statement of Theorem 2.11.

*Proof of Theorem 2.11.* We apply Lemma 7.1 to compute a sparsifier  $D - \widehat{M} \approx_{\varepsilon/16} D - M = B$  with  $\text{nnz}(\widehat{M}) \leq O(\varepsilon^{-2} n \log^c n)$  in work  $O(m_B \log^{c_1} n)$  and depth  $O(\log^{c_2} n)$ . Then by Fact 3.1.d we have  $(D - \widehat{M})^{-1} \approx_{\varepsilon/8} (D - M)^{-1}$ .

Our goal now is to construct a sparse  $\varepsilon$ -approximate inverse chain of the sparsifier  $\widehat{B} = D - \widehat{M}$ . The condition number of matrix  $\widehat{B}$  satisfies  $\kappa_{\widehat{B}} \leq \frac{1+\varepsilon/8}{1-\varepsilon/8} \kappa_B = t_{\widehat{B}}$ . Let  $\varepsilon' = \varepsilon/(16 \log t_{\widehat{B}})$ . Spielman and Peng [21] proved that  $O(\log t_{\widehat{B}})$  iterations suffice for the following iterative procedure to output a sparse  $\varepsilon$ -approximate inverse chain.

By Lemma 7.2 we compute a spectral sparsifier  $D - \widehat{M}_2 \approx_{\varepsilon/8} D - \widehat{M} D^{-1} \widehat{M}$  with  $\text{nnz}(\widehat{M}_2) \leq O(\varepsilon^{-2} n \log^c n)$  in work  $O(\varepsilon^{-4} n \log^{c+c_1+1} n)$  and depth  $O(\log^{c_2} n)$ . By Lemma 3.8 and Lemma 7.3 for each consecutive call we compute a sparsifier  $D - \widehat{M}_{2^{k+1}} \approx_{\varepsilon'} D - \widehat{M}_{2^k} D^{-1} \widehat{M}_{2^k}$  with the at most  $O(\varepsilon^{-2} n \log^c n \cdot \log^2 \kappa_B)$  non-zero entries in work  $O(\varepsilon^{-4} n \log^{c+c_1+1} n \cdot \log^2 \kappa_B)$  and depth  $O(\log^{c_2} n)$ .

We combine now Fact 3.1 and apply recursively  $O(\log t_{\widehat{B}})$  times the relation

$$\begin{aligned} & [D^{-1} + (I + D^{-1} \widehat{M}_{2^k}) [D - \widehat{M}_{2^{k+1}}]^{-1} (I + \widehat{M}_{2^k} D^{-1})] / 2 \\ & \approx_{\varepsilon'} [D^{-1} + (I + D^{-1} \widehat{M}_{2^k}) [D - \widehat{M}_{2^k} D^{-1} \widehat{M}_{2^k}]^{-1} (I + \widehat{M}_{2^k} D^{-1})] / 2 \\ & = (D - \widehat{M}_{2^k})^{-1}. \end{aligned}$$

Spielman and Peng showed in [21, Corollary 5.5] that  $D - \widehat{M}_{2^k} D^{-1} \widehat{M}_{2^k}$  can be replaced with  $D$  for  $k = O(\log t_{\widehat{B}})$  and maintain the desired approximation. The statement follows by Fact 3.1. ■

## 9 Representational Power of MDBD

In this section we prove Theorem 2.1. Our analysis relies on the following three influential works. Hald [12] analyzed mixed Binomial distributions in continuous case. Cruz-Urbe and Neugebauer [7, 8] gave sharp guarantees for approximating integrals using the Trapezoid method. Doha et al. [9] proved a simple closed formula for higher order derivatives of Bernstein basis.

Our goal now is to prove Theorem [2.1](#). Hald [\[12\]](#) proved the following result on mixed Binomial distributions.

**Theorem 9.1.** [\[12\]](#) *Let  $w(x)$  be a probability density function that is four times differentiable. Then for every  $N \in \mathbb{N}$  and  $i \in [0, N]$  the Bernstein basis  $B_{N,i}(p)$  satisfies*

$$\int_0^1 w(p) \cdot B_{N,i}(p) dp = \frac{w(i/N)}{N} \cdot \left[ 1 + \frac{b_1(i/N)}{N} + \frac{b_2(i/N)}{N^2} + O\left(\frac{1}{N^3}\right) \right] \quad (3)$$

where the functions are defined by  $b_1(x) = \frac{1}{w(x)}[-w(x) + (1-2x)w'(x) + \frac{1}{2}x(1-x)w''(x)]$  and  $b_2(x) = \frac{1}{w(x)}[w(x) - 3(1-2x)w'(x) + (1-6x+6x^2)w''(x) + \frac{5}{6}x(1-x)(1-2x)w'''(x) + \frac{1}{8}x^2(1-x)^2w^{(4)}(x)]$ .

We distinguish two types of approximation errors. The error term  $(1 + \eta_i)$  (c.f. Equation [1](#) in Theorem [2.1](#)) is caused by the error introduced in Equation [3](#). The second error type is due to the integral discretization with finite summation. The later approximation error is analyzed by Cruz-Uribe and Neugebauer [\[7, 8\]](#). We summarize below their result.

**Theorem 9.2.** [\[7, 8\]](#) *Suppose  $f$  be continuous and twice differentiable function,  $T \in \mathbb{N}$  is number, and the discrete approximator of  $f$  is defined by  $A_T(f) = [\frac{1}{2}(f(0) + f(1)) + \sum_{i=1}^{T-1} f(i/T)]/T$ . Then the approximation error is given by the expression  $E_T(f) = |A_T(f) - \int_0^1 f(t)dt| = |\sum_{i=1}^T L_i|$ , where  $L_i = \frac{1}{2} \int_{x_{i-1}}^{x_i} [\frac{1}{4T^2} - (t - c_i)^2] f''(t)dt$  and  $c_i = (x_{i-1} + x_i)/2$ .*

The rest of this section is devoted to prove Theorem [2.1](#). We use the following two results established by Cruz-Uribe and Neugebauer, and Doha et al.

**Lemma 9.3.** [\[7, 8\]](#) *The Bernstein basis satisfies  $\int_0^1 B_{N,i}(x)dx = \frac{1}{N+1}$  for every  $i \in [0 : N]$ .*

**Lemma 9.4.** [\[9\]](#) *The  $p$ th derivative of a Bernstein basis satisfies for every  $i \in [0 : N]$  that*

$$\frac{d^p B_{N,i}(x)}{dx^p} = \frac{N!}{(N-p)!} \sum_{k=\max\{0, i+p-N\}}^{\min\{i, p\}} (-1)^{k+p} \cdot \binom{p}{k} \cdot B_{N-p, i-k}(x).$$

We propose an upper bound on the integral of  $p$ th order derivative of Bernstein basis.

**Corollary 9.5.** *For every  $p \in [1 : N-1]$  and  $i \in [p+1 : N-(p+1)]$  such that  $i+p \leq N$ , the Bernstein basis satisfies*

$$\int_0^1 \left| \frac{d^p B_{N,i}(x)}{dx^p}(t) \right| dt \leq \frac{N!}{(N-p)!} \cdot \frac{2^p}{N+1}.$$

*Proof.* We combine Lemma [9.3](#) and Lemma [9.4](#) to obtain

$$\int_0^1 \left| \frac{d^p B_{N,i}(x)}{dx^p}(t) \right| dt \leq \frac{N!}{(N-p)!} \sum_{k=0}^p \binom{p}{k} \cdot \int_0^1 B_{N-p, i-k}(t) dt = \frac{N!}{(N-p)!} \cdot \frac{2^p}{N+1}.$$

■

We are now ready to prove Theorem [2.1](#).

*Proof of Theorem [2.1](#).* Recall that  $F_i(x) = w(x)B_{N,i}(x)$ . By Theorem [9.2](#) we have

$$\left| \sum_{k=1}^T L_k \right| = \left| \frac{1}{2} \sum_{k=1}^T \int_{x_{k-1}}^{x_k} \left[ \frac{1}{4T^2} - (t - c_k)^2 \right] \cdot \frac{d^2 F_i(x)}{dx^2}(t) dt \right| \leq \frac{1}{8T^2} \int_0^1 \left| \frac{d^2 F_i(x)}{dx^2}(t) \right| dt.$$

Since  $|\frac{d^2 F_i(x)}{dx^2}| = w'' \cdot B_{N,i} + 2 \cdot w' \cdot B'_{N,i} + w \cdot B''_{N,i}$ , we consider following three cases:

**Case 1:** We combine  $\max_{x \in [0,1]} |w''(x)| \leq 2\phi_w \cdot N^2$  and Lemma 9.3 to obtain

$$\int_0^1 |w''(t) \cdot B_{N,i}(t)| dt \leq 2\phi_w \cdot N.$$

**Case 2:** Using  $\max_{x \in [0,1]} |w'(x)| \leq \frac{1}{2}\phi_w \cdot N$  and Corollary 9.5 it holds

$$\int_0^1 |w'(t) \cdot B'_{N,i}(t)| dt \leq \phi_w \cdot N.$$

**Case 3:** Combining  $\max_{x \in [0,1]} |w(x)| \leq \phi_w$  and Corollary 9.5 yields

$$\int_0^1 |w(t) \cdot B''_{N,i}(t)| dt \leq \phi_w \cdot \int_0^1 B''_{N,i}(t) dt \leq 4\phi_w \cdot N.$$

The desired result follows from the preceding three cases and Theorem 9.1. ■

## 10 Approximating Discretized PDF

In this section we prove Theorem 2.2. We begin our discussion by presenting the pseudo code of algorithm **AppDscrPDF**.

---

**Algorithm 3** Approximate Discretized PDF by MDBD

---

$\mathcal{B}_{N,T}(\alpha, p) = \mathbf{AppDscrPDF}(w, \phi_w, N, \varepsilon_I)$

1. Compute  $S_{N+1,N} = \sum_{i=0}^N w(i/N)$  and  $S_{T,T+1} = \sum_{i=1}^T w(i/[T+1])$ , where  $T = \lceil N\sqrt{\phi_w/\varepsilon_I} \rceil$ .
  2. Compute  $\alpha_j = w(p_j) \cdot N/[(T+1) \cdot S_{N+1,N}]$ , where  $p_j = j/[T+1]$  for all  $j \in [1 : T]$ .
  3. Return  $\mathcal{B}_{N,T}(\alpha, p)$ .
- 

Before we prove Theorem 2.2, we analyze the class of continuous probability density functions that admit a discretized approximation by MDBD.

**Lemma 10.1.** *Let  $w(x) = C \cdot f(x)$  be a twice differentiable p.d.f. such that for  $N \in \mathbb{N}_+$  it holds a)  $0 \leq f(x) \leq 1$ , b)  $\frac{1}{2}[f(0) + f(1)] \geq \Omega(1)$ , c)  $1 \leq C \leq o(N)$ , and d)  $\int_0^1 |f^{(2)}(x)| dx \leq o(N)$ . Then, for  $T = \lceil N\sqrt{\phi_w/\varepsilon_I} \rceil$  with  $\phi_w/\varepsilon_I \geq 1$  it holds*

$$1 - \frac{S_{T,T+1}/(T+1)}{S_{N+1,N}/N} = o(1), \text{ where } S_{T,T+1} \triangleq \sum_{k=1}^T w(k/[T+1]) \text{ and } S_{N+1,N} \triangleq \sum_{k=0}^N w(k/N).$$

*Proof.* By Theorem 9.2 for the discrete approximator of  $f$

$$A_M(f) = \frac{1}{M} \left[ \frac{1}{2} (f(0) + f(1)) + \sum_{i=1}^{M-1} f\left(\frac{i}{M}\right) \right],$$

it holds that

$$\left| A_{T+1}(f) - \int_0^1 f(t) dt \right| \leq \frac{1}{8(T+1)^2} \int_0^1 |f^{(2)}(t)| dt \lesssim \frac{\varepsilon_I}{\phi_w} \cdot o\left(\frac{1}{N}\right),$$

and similarly

$$\left| A_N(f) - \int_0^1 f(t)dt \right| \leq \frac{1}{8N^2} \int_0^1 |f^{(2)}(t)| dt \lesssim o\left(\frac{1}{N}\right).$$

By definition,  $\int_0^1 f(t)dt = C^{-1} \in (1/o(N), 1]$  and thus

$$A_{T+1}(f) \in \left[ \frac{1}{C} \pm \frac{\varepsilon_I}{\phi_w} \cdot o\left(\frac{1}{N}\right) \right], \quad \text{and} \quad A_N(f) \in \left[ \frac{1}{C} \pm o\left(\frac{1}{N}\right) \right].$$

Let  $d \triangleq [f(0) + f(1)]/2$ . Straightforward checking shows that

$$\frac{S_{T,T+1}}{T+1} = C \cdot \left[ A_{T+1}(f) - \frac{d}{T+1} \right] \quad \text{and} \quad \frac{S_{N+1,N}}{N} = C \cdot \left[ A_N(f) + \frac{d}{N} \right].$$

We prove now the upper bound. By assumption  $d \in [\Omega(1), 1]$  and since  $C \leq o(N)$  we have

$$\begin{aligned} \Lambda &\triangleq \frac{S_{T,T+1}/(T+1)}{S_{N+1,N}/N} = \frac{A_{T+1}(f) - \frac{d}{T+1}}{A_N(f) + \frac{d}{N}} = \frac{\frac{1}{C} - \frac{d}{T+1} \pm \frac{\varepsilon_I}{\phi_w} \cdot o\left(\frac{1}{N}\right)}{\frac{1}{C} + \frac{d}{N} \pm o\left(\frac{1}{N}\right)} \\ &\leq 1 - \frac{\left(1 + \frac{1}{2}\sqrt{\frac{\varepsilon_I}{k}}\right) \cdot \frac{d}{N} - \left(1 + \frac{\varepsilon_I}{k}\right) \cdot o\left(\frac{1}{N}\right)}{\frac{1}{C} + \left[\frac{d}{N} - o\left(\frac{1}{N}\right)\right]} \leq 1 - \frac{\Omega(1)}{N} \cdot \frac{1}{\frac{1}{C} + \frac{1}{N}} = 1 - o(1). \end{aligned}$$

We can prove the lower bound  $\Lambda \geq 1 - o(1)$  using similar arguments. ■

We present now the proof of Theorem 2.2.

*Proof of Theorem 2.2.* By definition  $S_{T,T+1} = \sum_{k=1}^T w(j/[T+1])$ ,  $S_{N+1,N} = \sum_{j=0}^N w(j/N)$  and the desired discretized p.d.f. is

$$\hat{w}(i/N) = \frac{w(i/N)}{\sum_{j=0}^N w(j/N)} = \frac{w(i/N)}{S_{N+1,N}}.$$

By Theorem 2.1, it holds for all  $i \in [3 : N-3]$  that

$$\left| (1 + \eta_i) \frac{w(i/N)}{N} - \sum_{j=1}^T \frac{w(j/[T+1])}{T+1} \cdot B_{N,i}(j/[T+1]) \right| \leq \frac{\varepsilon_I}{N}. \quad (4)$$

We construct now MDBD  $\mathcal{B}_{N,T}(\alpha, p)$  as follows: for all  $j \in [1 : T]$  we set

$$\alpha_j = \frac{w(p_j) \cdot N}{S_{N+1,N} \cdot (T+1)}, \quad \text{where} \quad p_j = \frac{j}{T+1}.$$

Moreover,  $\mathcal{B}_{N,T}(\alpha, p)$  induces a vector  $\gamma$  that satisfies  $\gamma_i = \sum_{j=1}^T \alpha_j \cdot B_{N,i}(p_j)$  for all  $i \in [0 : N]$ . By multiplying Equation 4 with  $N/S_{N+1,N}$  we obtain

$$|(1 + \eta_i) \hat{w}(i/N) - \gamma_i| \leq \varepsilon_I / S_{N+1,N}. \quad (5)$$

Furthermore, since

$$\sum_{j=1}^T \alpha_j = \frac{N}{(T+1) \cdot S_{N+1,N}} \sum_{j=1}^T w(p_j) = \frac{S_{T,T+1}/(T+1)}{S_{N+1,N}/N},$$

by Lemma 10.1 there is a small positive number  $\delta_w = o(1)$  such that

$$\delta_w = 1 - \frac{S_{T,T+1}/(T+1)}{S_{N+1,N}/N} = 1 - \sum_{j=1}^T \alpha_j.$$

Hence, we have

$$\sum_{i=0}^N \gamma_i = \sum_{i=0}^N \sum_{j=1}^T \alpha_j \cdot B_{N,i}(p_j) = \sum_{j=1}^T \alpha_j \sum_{i=0}^N B_{N,i}(p_j) = \sum_{j=1}^T \alpha_j = 1 - \delta_w.$$

Since  $\delta_w = o(1)$ , by Equation 5 it follows that  $\gamma/[1 - \delta_w]$  is a discretized probability distribution over  $[0 : N]$  that approximates component-wise the desired discretized p.d.f.  $\hat{w}$ .

We note that the summations  $S_{N+1,N}$  and  $S_{T,T+1}$  can be computed in  $O(N + T)$  work and  $O(\log N + \log T)$  depth. Hence, the statement follows.  $\blacksquare$

## 11 Efficient Parallel Solver for Transpose Bernstein-Vandermonde Systems

**Problem 1.** Suppose a vector  $\gamma \in (0, 1)^{N+1}$  is induced by a convex combination of exactly  $N + 1$  discrete Binomial distributions  $B(p_i, N)$  such that  $0 < p_i \neq p_j < 1$  for all  $i \neq j$ . Find the unique vector  $\alpha \in (0, 1)^{N+1}$  such that  $\gamma_i = \sum_{j=1}^{N+1} \alpha_j \cdot B_{N,i}(p_j)$  for all  $i \in [0 : N]$ .

The Bernstein basis is a well studied primitive in the literature for polynomial interpolations [7, 8]. It is defined by  $B_{N,k}(p) = \binom{N}{k} p^k (1-p)^{N-k}$  for any  $k \in [0 : N]$ . Let  $\mathcal{B}_{N,T}(\alpha, p)$  be MDBD with  $T = N + 1$ . Then the Bernstein basis matrix is defined by  $[\mathbf{B}_N(p)]_{ji} = B_{N,i}(p_j)$ ,  $\forall i, j \in [1 : N + 1]$ , and it has a full rank (c.f. Appendix C). Moreover, the vector  $\alpha$  is the unique solution of the linear system  $\mathbf{B}_N(p)^T \alpha = \gamma$ .

In this section, we give an efficient parallel algorithm that solves Problem 1 and works in nearly linear work and poly-logarithmic depth. Our goal now is to prove Theorem 2.3. We reduce a transpose Bernstein-Vandermonde system to a transpose Vandermonde system that can be solved efficiently and in parallel by a variation of an algorithm proposed by Gohberg and Olshevsky [11]. We present now their main algorithmic result.

**Theorem 11.1.** [11] There is an algorithm that on input two vectors  $\alpha, p \in \mathbb{R}^{N+1}$  such that  $0 < p_i \neq p_j < 1$  for all  $i \neq j$ , outputs the vector  $\gamma = \mathbf{V}(p)^T \alpha$  in  $O(N \log^2 N)$  time.

Theorem 11.1 follows by [11, Algorithm 2.1] which relies on a non-trivial matrix decomposition of  $\mathbf{V}(p)^T$  to compute in  $O(N \log^2 N)$  time the desired matrix-vector product. It can be easily verified that [11, Algorithm 2.1] can be amended to compute the vector  $\alpha = [\mathbf{V}(p)^T]^{-1} \gamma$  in  $O(N \log^2 N)$  time. Furthermore, straightforward checking shows that this modified algorithm can be easily parallelized. We summarize below the resulting parallel algorithm.

**Theorem 11.2.** [11] There is a parallel algorithm that on input two vectors  $\gamma, p \in \mathbb{R}^{N+1}$  as in Theorem 11.1, outputs the vector  $\alpha = [\mathbf{V}(p)^T]^{-1} \gamma$  in  $O(N \log^2 N)$  work and  $O(\log^c n)$  depth, for some constant  $c \in \mathbb{N}_+$ .

We prove now that the Bernstein basis matrix  $\mathbf{B}_N(p)$  admits the following decomposition.

**Lemma 11.3.** Suppose  $p \in (0, 1)^{N+1}$  is vector such that  $0 < p_i \neq p_j < 1$  for all  $i \neq j$ ,  $\mathbf{V}(p)$  is Vandermonde matrix defined by  $[\mathbf{V}(p)]_{ji} = (\frac{p_j}{1-p_j})^i$ ,  $D_p = \text{diag}(\{(1-p_j)^N\}_{j=1}^{N+1})$  and  $D_{CN} = \text{diag}(\{\binom{N}{i}\}_{i=0}^N)$  are positive diagonal matrices. Then it holds that  $\mathbf{B}_N(p) = D_p \cdot \mathbf{V}(p) \cdot D_{CN}$ .

*Proof.* By definition  $[D_p \cdot \mathbf{V}(p) \cdot D_{CN}]_{j,i} = (1-p_j)^N (\frac{p_j}{1-p_j})^i \binom{N}{i} = B_{N,i}(p_j) = [\mathbf{B}_N(p)]_{ji}$ . ■

We are ready now to prove Theorem 2.3.

*Proof of Theorem 2.3.* By Lemma C.1 the Bernstein matrix  $\mathbf{B}_N(p)$  is invertible. Given a vector  $\gamma \in (0, 1)^{N+1}$  we want to find the vector  $\alpha = [\mathbf{B}_N(p)^T]^{-1}\gamma$ . By Lemma 11.3 we have  $[\mathbf{B}_N(p)^T]^{-1} = [D_p]^{-1} \cdot [\mathbf{V}(p)^T]^{-1} \cdot [D_{CN}]^{-1}$ . Moreover, we can compute a vector  $\gamma' = [D_{CN}]^{-1}\gamma$  in  $O(n)$  time. Using Theorem 11.2, we obtain a vector  $\gamma'' = [\mathbf{V}(p)^T]^{-1}\gamma'$  in  $O(N \log^2 N)$  work and  $O(\log^c n)$  depth. The desired vector  $\alpha = [D_p]^{-1}\gamma''$  takes further  $O(n)$  work and  $O(1)$  depth to compute. ■

**Acknowledgements** We are grateful to Arijit Ghosh and Kunal Dutta for the helpful discussions on mixture of Binomial distributions, and to Arnur Nigmatov and Shay Moran for pointing us to the Bernstein interpolation polynomials. We would also like to thank Kurt Mehlhorn for the insightful comments and suggestions to the early version of the manuscript.

This work has been funded by the Cluster of Excellence “Multimodal Computing and Interaction” within the Excellence Initiative of the German Federal Government.

## References

- [1] R. Andersen, F. R. K. Chung, and K. J. Lang. Local graph partitioning using pagerank vectors. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, 21-24 October 2006, Berkeley, California, USA, *Proceedings*, pages 475–486, 2006. URL <http://dx.doi.org/10.1109/FOCS.2006.44>.
- [2] M. W. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo. Support-graph preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 27(4):930–951, 2006. doi: 10.1137/S0895479801384019. URL <http://dx.doi.org/10.1137/S0895479801384019>.
- [3] D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S. Teng. Scalable parallel factorizations of SDD matrices and efficient sampling for gaussian graphical models. *CoRR*, abs/1410.5392, 2014. URL <http://arxiv.org/abs/1410.5392>.
- [4] D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S. Teng. Efficient sampling for gaussian graphical models via spectral sparsification. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 364–390, 2015. URL <http://jmlr.org/proceedings/papers/v40/Cheng15.html>.
- [5] D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S. Teng. Spectral sparsification of random-walk matrix polynomials. *CoRR*, abs/1502.03496, 2015. URL <http://arxiv.org/abs/1502.03496>.
- [6] M. B. Cohen, Y. T. Lee, C. Musco, C. Musco, R. Peng, and A. Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 181–190, 2015. doi: 10.1145/2688073.2688113. URL <http://doi.acm.org/10.1145/2688073.2688113>.



- [7] D. Cruz-Uribe and C. Neugebauer. Sharp error bounds for the trapezoidal rule and Simpson’s rule. *JIPAM. J. Inequal. Pure Appl. Math.*, 3(49), 2002. URL [http://www.emis.de/journals/JIPAM/images/031\\_02\\_JIPAM/031\\_02.pdf](http://www.emis.de/journals/JIPAM/images/031_02_JIPAM/031_02.pdf).
- [8] D. Cruz-Uribe and C. J. Neugebauer. An elementary proof of error estimates for the trapezoidal rule. *Mathematics Magazine*, 76(4):pp. 303–306, 2003. ISSN 0025570X. URL <http://www.jstor.org/stable/3219088>.
- [9] E. Doha, A. Bhrawy, and M. Saker. On the derivatives of Bernstein polynomials: an application for the solution of high even-order differential equations. *Bound. Value Probl.*, 2011:16, 2011. ISSN 1687-2770/e. doi: 10.1155/2011/829543.
- [10] S. O. Gharan and L. Trevisan. Approximating the expansion profile and almost optimal local graph clustering. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 187–196, 2012. URL <http://dx.doi.org/10.1109/FOCS.2012.85>.
- [11] I. Gohberg and V. Olshevsky. Fast algorithms with preprocessing for matrix-vector multiplication problems. *Journal of Complexity*, 10(4):411 – 427, 1994. ISSN 0885-064X. URL <http://www.sciencedirect.com/science/article/pii/S0885064X84710211>.
- [12] A. Hald. The mixed binomial distributin and the posterior distribution of p for a continuous prior distribution. *Journal of the Royal Statistical Society. Series B (Methodological)*, 30(2): pp. 359–367, 1968. ISSN 00359246. URL <http://www.jstor.org/stable/2984516>.
- [13] J. A. Kelner and A. Levin. Spectral sparsification in the semi-streaming setting. *Theory Comput. Syst.*, 53(2):243–262, 2013. URL <http://dx.doi.org/10.1007/s00224-012-9396-1>.
- [14] P. Kolev and H. Sun. Dirichlet eigenvalues, local random walks, and analyzing clusters in graphs. In *Algorithms and Computation - 25th International Symposium, ISAAC 2014, Jeonju, Korea, December 15-17, 2014, Proceedings*, pages 621–632, 2014. URL [http://dx.doi.org/10.1007/978-3-319-13075-0\\_49](http://dx.doi.org/10.1007/978-3-319-13075-0_49).
- [15] I. Koutis, G. L. Miller, and R. Peng. A nearly-m log n time solver for sdd linear systems. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science, FOCS ’11*, pages 590–598, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4571-4. URL <http://dx.doi.org/10.1109/FOCS.2011.85>.
- [16] I. Koutis, A. Levin, and R. Peng. Improved spectral sparsification and numerical algorithms for SDD matrices. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, pages 266–277, 2012. URL <http://dx.doi.org/10.4230/LIPIcs.STACS.2012.266>.
- [17] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. *CoRR*, abs/1512.01892, 2015. URL <http://arxiv.org/abs/1512.01892>.
- [18] G. L. Miller and R. Peng. Approximate maximum flow on separable undirected graphs. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1151–1170, 2013. URL <http://dx.doi.org/10.1137/1.9781611973105.83>.

- [19] H. Minc. *Nonnegative Matrices*. John Wiley and Sons, New York, 1973.
- [20] L. Orecchia and N. K. Vishnoi. Towards an sdp-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 532–545. SIAM, 2011. URL <http://dl.acm.org/citation.cfm?id=2133036.2133078>.
- [21] R. Peng and D. A. Spielman. An efficient parallel solver for SDD linear systems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 333–342, 2014. URL <http://doi.acm.org/10.1145/2591796.2591832>.
- [22] R. Y. Peng. *Algorithm Design Using Spectral Graph Theory*. PhD thesis, Carnegie Mellon University, Pittsburgh, August 2013. CMU CS Tech Report CMU-CS-13-121.
- [23] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Inf. Comput.*, 82(1):93–133, 1989. URL [http://dx.doi.org/10.1016/0890-5401\(89\)90067-9](http://dx.doi.org/10.1016/0890-5401(89)90067-9).
- [24] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 563–568, 2008. URL <http://doi.acm.org/10.1145/1374376.1374456>.
- [25] D. A. Spielman and S. Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4): 981–1025, 2011. doi: 10.1137/08074489X. URL <http://dx.doi.org/10.1137/08074489X>.
- [26] D. A. Spielman and S.-H. Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014. URL <http://dx.doi.org/10.1137/090771430>.

## A Generalized Escaping Probability

*Proof of Lemma 2.10.* By definition, for every vector  $x$  the spectral sparsifier  $D - \hat{A}$  preserves approximately the quadratic form

$$x^T(D - \hat{A})x \in [(1 \pm \varepsilon) \cdot x^T(D - D\mathcal{G}_\gamma)x].$$

Hence, the statement follows by applying the identities

$$\begin{aligned} \xi_S^T(D - D\mathcal{G}_\gamma)\xi_S &= \pi_S^T(I - \mathcal{G}_\gamma)\mathbf{1}_S = 1 - \pi_S^T\mathcal{G}_\gamma\mathbf{1}_S = \pi_S^T\mathcal{G}_\gamma\mathbf{1}_{\bar{S}} \\ &= \mathbb{E}_{v \sim \pi_S} [\text{gEsc}(v, S, \mathcal{G}_\gamma)]. \end{aligned}$$

■

## B Spectral Sparsification of $\mathcal{T}$ -Matrices

Our proof of Lemma B.2 is based on the Perron-Frobenius Theorem [19] for non-negative matrices.

**Theorem B.1.** [19, Perron-Frobenius] *Suppose  $A$  is symmetric nonnegative matrix. Then it has a nonnegative eigenvalue  $\lambda$  which is greater than or equal to the modulus of all other eigenvalues.*

**Lemma B.2.** Suppose  $D - M$  is SDDM matrix. Then  $D - D(D^{-1}M)^N$  is SDDM matrix  $\forall N \in \mathbb{N}_+$ .

*Proof.* Since  $D - M$  is SDDM we have  $D \succ M$ . By Fact 3.1.e it holds  $I \succ D^{-1/2}MD^{-1/2} \triangleq X$ . Hence, the largest eigenvalue  $\lambda(X) < 1$ . By Theorem B.1 the spectral radius  $\rho(X) < 1$ , i.e.  $|\lambda_i(X)| < 1$  for all  $i$ . Since  $X$  is symmetric it has the form  $X = \sum_i \lambda_i u_i u_i^T$ . Moreover, we have  $X^k = \sum_i \lambda_i^k u_i u_i^T$  for every  $k \in \mathbb{N}_+$ . Thus the spectral radius of  $X^k$  satisfies  $\rho(X^k) = \rho(X)^k < 1$ .

Notice that  $B_k = D - D(D^{-1}M)^k$  is symmetric non-negative matrix for every  $k \in \mathbb{N}_+$ . By definition  $D - M$  is diagonally dominant and thus  $D^{-1}M\mathbf{1} \preceq \mathbf{1}$  component-wise. This implies that  $B_k$  is diagonally dominant matrix. Notice that  $B_k = D^{1/2}[I - X^k]D^{1/2}$  and since  $\rho(X^k) < 1$  it follows that  $B_k$  is positive definite and hence SDDM matrix. ■

**Proof of Lemma 3.8** We combine Lemma B.2 with the following two statements.

**Fact B.3.** Spielman and Peng [21, Proposition 5.6] showed that if  $D - M$  is SDDM matrix, then  $D - MD^{-1}M$  is SDDM matrix. Also Cheng et al. [4, Proposition 25] showed that if  $D - M$  is Laplacian matrix then  $D - D(D^{-1}M)^N$  is Laplacian matrix for every  $N \in \mathbb{N}_+$ .

Based on Lemma 3.9 and Fact B.3 we establish the following result.

**Lemma B.4.** Suppose  $D - M$  is  $\mathcal{T}$ -matrix and  $D - \widehat{M} \approx_\varepsilon D - M$  is a spectral sparsifier. Then  $D - D(D^{-1}\widehat{M})^N$  is  $\mathcal{T}$ -matrix for every  $N \in \mathbb{N}_+$ .

**Proof of Lemma 3.7** We use the following result that appears in Peng's thesis [22].

**Lemma B.5.** [22] Suppose  $D - A$  is Laplacian matrix (possibly  $A_{ii} \neq 0$ ), and  $\widetilde{D} - \widetilde{A}$  a sparsifier with  $\widetilde{A}_{ii} = 0$  for every  $i$  such that  $(1 - \varepsilon)(D - A) \preceq \widetilde{D} - \widetilde{A} \preceq D - A$ . Then the symmetric non-negative matrix  $\widehat{A} = (D - \widetilde{D}) + \widetilde{A}$  satisfies  $(1 - \varepsilon)(D - A) \preceq D - \widehat{A} \preceq (D - A)$ .

*Proof of Lemma 3.7.* Let  $\widetilde{D}_1 = \frac{1}{1+\varepsilon}\widetilde{D}$  and  $\widetilde{A}_1 = \frac{1}{1+\varepsilon}\widetilde{A}$ . Then  $\frac{1-\varepsilon}{1+\varepsilon}(D - A) \preceq \widetilde{D}_1 - \widetilde{A}_1 \preceq D - A$  and by Lemma B.5 the symmetric non-negative matrix  $\widehat{A} = (D - \widetilde{D}_1) + \widetilde{A}_1$  satisfies  $\frac{1-\varepsilon}{1+\varepsilon}(D - A) \preceq D - \widehat{A} \preceq D - A$ . Since  $\frac{1-\varepsilon}{1+\varepsilon} \geq 1 - 2\varepsilon$  for every  $\varepsilon \in (0, \frac{1}{2})$  the statement follows. ■

## B.1 Structural Result

Suppose  $D - M$  is  $\mathcal{T}$ -matrix. We show that the matrix  $D - MD^{-1}M$  can be expressed as a sum of a non-negative main diagonal matrix and a sum of Laplacian matrices.

*Proof of Lemma 3.10.* Let  $M \in \mathbb{R}^{n \times n}$  and  $nnz(M) = m$ . We decompose the entries of matrix  $MD^{-1}M$  into three types. We set type 1 to be the entries  $(MD^{-1}M)_{ii} = \sum_{k=1}^n M_{ik}^2/D_k$  for all  $i$ . We note that all entries of type 1 can be computed in  $O(m)$  time. We consider next the off-diagonal entries

$$(MD^{-1}M)_{ij} = \underbrace{(M_{ii}/d_i + M_{jj}/d_j) \cdot M_{ij}}_{\text{type 2}} + \underbrace{\sum_{k \neq \{i,j\}}^n M_{ik}M_{jk}/D_k}_{\text{type 3}}.$$

Observe that the number of type 2 entries is at most  $m$ . Now for a fixed  $k$  we note that the corresponding entries that appear in type 1 and type 3 form a weighted clique (with self-loops) whose adjacency matrix is defined by  $\frac{1}{d_k}\eta_k\eta_k^T$ .

Straightforward checking shows that  $MD^{-1}M = \mathbf{B} + \sum_i \frac{1}{d_i}\eta_i\eta_i^T$ . By Lemma 3.8  $D - MD^{-1}M$  is  $\mathcal{T}$ -matrix and thus diagonally dominant. Hence, the Laplacian matrices  $\mathbf{L}_B$  and  $\mathbf{L}_{N_i}$  for all  $i$

exist. Moreover, we can compute in  $O(m)$  time the positive diagonal matrices  $\mathbf{D}_B$  and  $\mathbf{D}_{N_i} = (s_i/d_i)\text{diag}(N_i)$  for all  $i$ . To see this, observe that  $s_i$  and  $d_i$  can be computed in  $O(m)$  time for all  $i$ , and the number of elements in the disjoint union  $\sqcup_i N_i \leq m$ . ■

## C Bernstein Basis Matrix

We prove below that the Bernstein basis matrix in Problem 1 has full rank.

**Lemma C.1.** *Suppose a vector  $p \in (0, 1)^{N+1}$  satisfies  $0 < p_i \neq p_j < 1$  for all  $i \neq j$ . Then the Bernstein basis matrix  $\mathbf{B}_N(p)$  has a full rank.*

*Proof.* Suppose for contradiction that  $\text{rank}(\mathbf{B}_N(p)) < N + 1$ . Then there is a vector  $\lambda \in \mathbb{R}^{N+1}$  such that the linear combination of the columns of  $\mathbf{B}_N(p)$  satisfies  $\sum_{i=0}^N \lambda_j [\mathbf{B}_N(p)]_{:,i} = 0$ . Let  $f_\lambda(x)$  be a polynomial defined by  $f_\lambda(x) \triangleq \sum_{i=0}^N \lambda_i \cdot B_{N,i}(x) = \sum_{i=0}^N \lambda_i \cdot \binom{N}{i} x^i (1-x)^{N-i}$ . Notice that  $f_\lambda(p_j) = 0$  for every  $j \in [1 : N + 1]$ , i.e.  $f_\lambda(x)$  has  $N + 1$  roots. However, since the polynomial  $f_\lambda(x)$  has degree  $N$  it follows that  $f_\lambda(x) \equiv 0$ . Therefore, we obtained the desired contradiction. ■

## D Approximating Two Canonical PDFs

Here, we illustrate the representational power of MDBD. We show that there are MDBD satisfying the hypothesis in Theorem 2.2 and approximate two canonical continuous p.d.f.: the Uniform distribution and the Exponential Families. More precisely, we prove that the Uniform distribution and the Exponential families admit a multiplicative and an additive approximation, respectively.

### D.1 Uniform Distribution

**Lemma D.1** (Uniform Distribution). *Let  $w(x) = 1$ ,  $N \in \mathbb{N}_+$  and  $\varepsilon > 0$ . If  $T \geq \Omega(N\varepsilon^{-1/2})$  then it holds that*

$$\frac{1}{T+1} \sum_{j=1}^T B_{N,i} \left( \frac{j}{T+1} \right) \in \left[ (1 \pm \varepsilon) \frac{1}{N+1} \right], \text{ for all } i \in [3 : N-3].$$

*Proof.* By Theorem 9.2 we have that

$$\left| \sum_{i=1}^T L_i \right| \leq \frac{1}{2} \sum_{i=1}^T \int_{x_{i-1}}^{x_i} \left| \frac{1}{4N^2} - (t - c_i)^2 \right| \left| \frac{d^2 B_{N,i}(x)}{dx^2}(t) \right| dt \leq \frac{1}{8N^2} \int_0^1 \left| \frac{d^2 B_{N,i}(x)}{dx^2}(t) \right| dt.$$

By combining Lemma 9.3 and Corollary 9.5 for every  $i \in [3 : N-3]$  it holds

$$\left| \frac{1}{N+1} - \frac{1}{T+1} \sum_{j=1}^T B_{N,i} \left( \frac{j}{T+1} \right) \right| \leq \left| \sum_{i=1}^T L_i \right| \leq \frac{1}{8T^2} \int_0^1 \left| \frac{d^2 B_{N,i}(x)}{dx^2}(x) \right| dx \leq \frac{N}{2T^2}.$$

We note that  $\frac{N}{2T^2} \leq \frac{\varepsilon}{N+1}$  since  $T \geq \Omega(N\varepsilon^{-1/2})$ , and hence the statement follows. ■

**Remark D.2.** *All conditions of Theorem 2.2 hold. Note that  $w(x) = 1 \cdot 1$ , i.e.,  $f(x) = 1$ . a)  $C = 1$ , b)  $f(x) = 1$ , c)  $\frac{1}{2}[f(0) + f(1)] = 1$  and d)  $\int_0^1 |f^{(2)}(x)| dx = 0$ , since  $f^{(2)}(x) = 0$ .*

## D.2 Exponential Families

**Lemma D.3** (Exponential Families). *Let  $N \in \mathbb{N}$ ,  $k \in [1, \sqrt{N}]$  and  $w(x) = \frac{k}{1-e^{-k}} \cdot \exp\{-k \cdot x\}$  is a probability density function. For any  $\varepsilon > 0$  if  $T \geq \Omega(N\sqrt{k/\varepsilon})$  then for every  $i \in [3 : N-3]$  it holds for the function  $F_i(x) = w(x) \cdot B_{N,i}(x)$  that*

$$\left| (1 + \eta_i) \frac{w(i/N)}{N} - \frac{1}{T+1} \sum_{j=1}^T F_i\left(\frac{j}{T+1}\right) \right| \leq \frac{\varepsilon}{N}, \quad \text{where } |\eta_i| \leq \frac{1}{4}.$$

*Proof.* The  $p$ th derivative of function  $w(x)$  satisfies  $w^{(p)}(x) = (-k)^p \cdot w(x)$ . Let  $I = [0, 1]$  be an interval. Straightforward checking shows that

$$\max_{x \in I} |w^{(p)}(x)| = k^p \cdot \max_{x \in I} |w(x)| < 2 \cdot k^{p+1}. \quad (6)$$

By the definition of function  $b_1(x)$  (c.f. Theorem 9.1) we have

$$b_1(x) = -\frac{k^2}{2} \cdot x^2 + \left(2k + \frac{k^2}{2}\right) \cdot x - (1 + k),$$

and we can show that  $\max_{x \in I} b_1(x) \leq 1 + k^2/8 \leq 1 + N/8$ . The function  $b_2(x)$  satisfies

$$b_2(x) = 1 + 3(1 - 2x)k + (1 - 6x + 6x^2)k^2 - \frac{5}{6}x(1 - x)(1 - 2x)k^3 + \frac{1}{8}x^2(1 - x)^2k^4,$$

and we can show that  $\max_{x \in I} b_2(x) \ll k^4/8 \leq N^2/8$ . By Theorem 2.1 it suffices to upper bound the following four cases.

**Case 1:** By Theorem 9.1  $\mu \leq \frac{1}{4}$ , since  $\max_{x \in I} |b_1(x)| \leq 1 + N/8$  and  $\max_{x \in I} |b_2(x)| \ll N^2/8$ .

**Case 2:** We combine Equation 6 and Lemma 9.3 to obtain

$$\int_0^1 |w''(t) \cdot B_{N,i}(t)| dt < 2 \cdot k^3 \cdot \int_0^1 |B_{N,i}(t)| dt = \frac{2 \cdot k^3}{N+1}.$$

**Case 3:** By combining Equation 6 and Lemma 9.5 it holds

$$\int_0^1 |w'(t) \cdot B'_{N,i}(t)| dt < 2 \cdot k^2 \cdot \int_0^1 |B'_{N,i}(t)| dt < 4 \cdot k^2.$$

**Case 4:** We use again Equation 6 and Lemma 9.5 to obtain

$$\int_0^1 |w(t) \cdot B''_{N,i}(t)| dt < 2 \cdot k \cdot \int_0^1 |B''_{N,i}(t)| dt < 8 \cdot k \cdot N.$$

Recall that  $F_i(x) = w(x)B_{N,i}(x)$ . By combining  $\left|\frac{d^2 F_i(x)}{dx^2}\right| = w'' \cdot B_{N,i} + 2w' \cdot B'_{N,i} + w \cdot B''_{N,i}$  and Theorem 9.2 we have

$$\left| \sum_{i=1}^T L_i \right| \leq \frac{1}{8T^2} \int_0^1 \left| \frac{d^2 F(x)}{dx^2}(t) \right| dt \lesssim \frac{k \cdot N}{T^2}.$$

Hence, the statement follows. ■

**Remark D.4.** The hypothesis in Theorem 2.2 holds. Note that  $w(x) = \frac{k}{1-e^{-k}} \cdot \exp\{-k \cdot x\}$  and  $\phi_w = \frac{k}{1-e^{-k}}$ . Thus  $C = \frac{k}{1-e^{-k}}$ ,  $f(x) = \exp\{-k \cdot x\}$ ,  $f^{(2)}(x) = k^2 \cdot f(x)$  and  $k \in [1, \sqrt{N}]$ . Furthermore,  
a)  $C = o(N)$ , b)  $0 \leq f(x) \leq 1$ , c)  $\frac{1}{2} \leq \frac{1}{2}[f(0) + f(1)] < 1$  and for d) we have

$$\int_0^1 |f^{(2)}(x)| dx = k^2 \int_0^1 |f(x)| dx = \frac{k^2}{C} = (1 - e^{-k})k = o(N).$$

## E Schur Complement

In this section we prove Lemma 4.1. We use the following result proposed by Peng et al. [5].

**Lemma E.1.** [5, Lemma 4.3] If  $M$  is SPSPD matrix and  $(1-\varepsilon)(D-M) \preceq D - \widehat{M} \preceq (1+\varepsilon)(D-M)$  then it holds that  $(1-\varepsilon)(D+M) \preceq D + \widehat{M} \preceq (1+\varepsilon)(D+M)$ .

We extend next two technical results on Schur complement that appeared in [4, 18, 22].

**Claim E.2.** Suppose  $X \triangleq \begin{bmatrix} P_1 & -M \\ -M & P_2 \end{bmatrix}$  where  $P_1$  and  $P_2$  are symmetric positive definite matrices and  $M$  is symmetric matrix. Then  $v^T[P_2 - MP_1^{-1}M]v = \min_u \begin{pmatrix} u \\ v \end{pmatrix}^T X \begin{pmatrix} u \\ v \end{pmatrix}$  for every  $v$ .

*Proof.* Suppose  $f_v(u) \triangleq \begin{pmatrix} u \\ v \end{pmatrix}^T \begin{bmatrix} P_1 & -M \\ -M & P_2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = u^T P_1 u - 2u^T M v + v^T P_2 v$ . Notice that  $f$  is minimized when  $u = P_1^{-1} M v$ , since  $\nabla f_v(u) = 2P_1 u - 2M v$ . Hence, it follows that  $\min_u \begin{pmatrix} u \\ v \end{pmatrix}^T X \begin{pmatrix} u \\ v \end{pmatrix} = v^T P_2 v - v^T M P_1^{-1} M v = v^T [P_2 - M P_1^{-1} M] v$ . ■

**Lemma E.3.** (Schur Complement) Suppose  $D_1, D_2$  are positive main diagonal matrices and  $M, Q$  are symmetric matrices. If  $X_M \triangleq \begin{bmatrix} D_1 & -M \\ -M & D_2 \end{bmatrix} \approx_\varepsilon \begin{bmatrix} D_1 & -Q \\ -Q & D_2 \end{bmatrix} \triangleq X_Q$  then it holds that  $D_2 - M D_1^{-1} M \approx_\varepsilon D_2 - Q D_1^{-1} Q$ .

*Proof.* Here we show the upper bound, but the lower bound follows by analogy. Let  $w$  be a vector such that  $\begin{pmatrix} w \\ v \end{pmatrix}^T X_Q \begin{pmatrix} w \\ v \end{pmatrix} = \min_u \begin{pmatrix} u \\ v \end{pmatrix}^T X_Q \begin{pmatrix} u \\ v \end{pmatrix}$ . We apply twice Claim E.2 to obtain the following chain of inequalities

$$\begin{aligned} v^T [D_2 - M D_1^{-1} M] v &= \min_u \begin{pmatrix} u \\ v \end{pmatrix}^T X_M \begin{pmatrix} u \\ v \end{pmatrix} \leq \begin{pmatrix} w \\ v \end{pmatrix}^T X_M \begin{pmatrix} w \\ v \end{pmatrix} \\ &\leq (1+\varepsilon) \begin{pmatrix} w \\ v \end{pmatrix}^T X_Q \begin{pmatrix} w \\ v \end{pmatrix} = (1+\varepsilon) \min_u \begin{pmatrix} u \\ v \end{pmatrix}^T X_Q \begin{pmatrix} u \\ v \end{pmatrix} \\ &= (1+\varepsilon) v^T [D_2 - Q D_1^{-1} Q] v. \end{aligned}$$

■

We prove Lemma 4.1 by arguing in a similar manner to in [5, Lemma 4.4]. We present the proof here for completeness.



*Proof of Lemma 4.1.* For any symmetric matrix  $X$ , we denote by  $\mathcal{P}_X = \begin{bmatrix} D & -X \\ -X & D \end{bmatrix}$ . We prove the upper bound, but the lower bound follows by analogy. Straightforward checking shows that

$$\begin{aligned} \begin{pmatrix} u \\ v \end{pmatrix}^T \mathcal{P}_{\widehat{M}} \begin{pmatrix} u \\ v \end{pmatrix} &= u^T D u - v^T \widehat{M} u - u^T \widehat{M} v + v^T D v \\ &= \frac{1}{2} \left[ (u+v)^T (D - \widehat{M}) (u+v) + (u-v)^T (D + \widehat{M}) (u-v) \right] \end{aligned}$$

By Lemma E.1 it holds that  $D + \widehat{M} \approx_\varepsilon D + M$  and thus we have

$$\begin{aligned} \begin{pmatrix} u \\ v \end{pmatrix}^T \mathcal{P}_{\widehat{M}} \begin{pmatrix} u \\ v \end{pmatrix} &= \frac{1}{2} \left[ (u+v)^T (D - \widehat{M}) (u+v) + (u-v)^T (D + \widehat{M}) (u-v) \right] \\ &\leq (1+\varepsilon) \frac{1}{2} \left[ (u+v)^T (D - M) (u+v) + (u-v)^T (D + M) (u-v) \right] \\ &= (1+\varepsilon) \begin{pmatrix} u \\ v \end{pmatrix}^T \mathcal{P}_M \begin{pmatrix} u \\ v \end{pmatrix}. \end{aligned}$$

Hence, it follows that  $\mathcal{P}_{\widehat{M}} = \begin{bmatrix} D & -\widehat{M} \\ -\widehat{M} & D \end{bmatrix} \approx_\varepsilon \begin{bmatrix} D & -M \\ -M & D \end{bmatrix} = \mathcal{P}_M$ . Now, by Lemma E.3 it holds that  $D - \widehat{M} D^{-1} \widehat{M} \approx_\varepsilon D - M D^{-1} M$ . ■